UNIVERSITAT POLITÈCNICA DE CATALUNYA

DEPARTAMENTO DE ORGANIZACIÓN DE EMPRESAS

Programa de Doctorado en Administración y Dirección de Empresas

MODELADO Y RESOLUCIÓN DE PROBLEMAS DE SECUENCIACIÓN EN CONTEXTO *JIT/DS* MEDIANTE *BDP*

DOCTORANDO:

ALBERTO CANO PÉREZ

TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR POR LA UNIVERSITAT POLITÈCNICA DE CATALUNYA

DIRECTOR DE TESIS:

JOAQUÍN BAUTISTA VALHONDO

Barcelona, Febrero de 2014

Agradecimientos

A mi familia y amigos por el apoyo que he recibido todos estos años, y especialmente a Gemma, la mujer de mi vida, por el apoyo y soporte moral en estos momentos tan duros.
También quiero agradecer la confianza y apoyo a mi jefe y tutor de tesis, Joaquín Bautista, sin el que todo este trabajo no hubiera sido posible. Por último, mención a mis compañeras de trabajo, Cristina y Rocío.

Alberto

Resumen

Este documento desarrolla la tesis doctoral titulada ***"Modelado y resolución de problemas de secuenciación en contexto JIT/DS mediante BDP"***. A través de una revisión de la literatura se explican los problemas objeto de estudio, enmarcándolo dentro del entorno de la secuenciación en los sistemas productivos. A continuación se describen dichos problemas, el *MMSP-W* (Mixed-Model Sequencing Problem with Work overload Minimization), el *BFSP* (Blocking Flow Shop Problem) y el *ORV* (Output Rate Variation Problem). Finalmente se proponen algoritmos de resolución para dichos problemas basados en *BDP* (Bounded Dynamic Programming o programación dinámica acotada) y se compara la calidad de los procedimientos propuestos con otros basados en programación lineal o heurísticas.

Keywords: Computer Science; Operations Research; Operations Management; Mixed-Model Sequencing Problem; Just-In-Time; Douki-Seisan; Dynamic Programming; Mixed Integer Linear Programming; Heuristics.

Abstract

This document develops the doctoral thesis entitled ***"Modelado y resolución de problemas de secuenciación en contexto JIT/DS mediante BDP"***. Through a literature review we explain the problems, within the environment of the sequence problems in production systems. Then, we describe the problems under study, the *MMSP-W* (Mixed-Model Sequencing Problem with Work overload Minimization), the *BFSP* (Blocking Flow Shop Problem) and the *ORV* (Output Rate Variation Problem). Then, a resolution procedure based on *BDP* (Bounded Dynamic Programming) to solve the problems is proposed. Finally, the results of the proposed procedure are compared with others based on for example linear programming or heuristics.

Keywords: Computer Science; Operations Research; Operations Management; Mixed-Model Sequencing Problem; Just-In-Time; Douki-Seisan; Dynamic Programming; Mixed Integer Linear Programming; Heuristics.

Índice General

Índice de tablas

Índice de figuras

Capítulo 1

Marco general de los problemas de secuenciación en el contexto *JIT* y *DS*

1.1. Introducción

En el entorno económico actual, el éxito y la eficiencia de cualquier empresa de producción están relacionados con los recursos disponibles y, sobre todo, con el modo de utilizarlos. El problema de cada empresa no acaba después de haber conseguido los medios de producción necesarios, ya que también tiene que resolver la cuestión de cómo explotar eficientemente dichos recursos. La programación de operaciones nació para resolver estas cuestiones de forma eficaz.

Definir un plan de producción optimizado es un problema importante y frecuente, ya que es necesario revisar las decisiones cada vez que se modifica la línea productiva o el programa de producción. En el contexto actual de crisis financiera, es esencial la revisión y mejora constante en el proceso y los productos con el objetivo de conseguir el mejor posicionamiento de la empresa en el mercado.

Antes de la aparición de algunas herramientas para la resolución de los problemas asociados a la programación de operaciones, las empresas resolvían éstos de forma manual e intuitiva, usando criterios basados en la experiencia adquirida por su personal. Esta forma de trabajo, aunque era muy flexible, no presentaba una base objetiva y no garantizaba una buena calidad de la solución escogida. En este contexto surge la necesidad de crear herramientas que proporcionen a estos problemas, si no una solución óptima, soluciones de calidad en un tiempo computacional adecuado. La realización de un plan de producción requiere esencialmente contestar a la pregunta: *"¿Qué operación se realiza en cada máquina en un determinado momento?"*.

Una vez realizados la planificación y el cálculo de necesidades y prosiguiendo en la adopción de decisiones cada vez con mayor nivel de detalle, debe procederse a la programación de operaciones: cada una de las tareas elementales que conducen a la realización de un programa maestro de producción debe asignarse a un recurso disponible de forma que se puedan realizar las órdenes

de trabajo emitidas y concretando los instantes (fechas) de dicha ejecución.
Programas muy detallados son comunes en diversos sectores industriales, de los cuales los horarios de ferrocarriles son uno de los ejemplos más conocidos. Cabe señalar que, si bien en algunos países el cumplimiento de los horarios es sólo aproximado, en otros la existencia de conexiones habituales entre diferentes líneas con márgenes bastante reducidos implica la confianza en que dicho cumplimiento alcanza un elevado grado de exactitud, lo que no supone que excepcionalmente no se produzcan incidencias y por tanto retrasos e incumplimientos. La regla es que el programa se cumpla; la excepción, que con frecuencia reducida se produzcan desviaciones. El programa marca la pauta a seguir y garantiza la sincronización de las actividades en la medida en que las desviaciones de la realidad respecto al programa sean pequeñas.
La programación, dada su proximidad a la realización y el nivel de concreción perseguido, no puede utilizar valores medios, tan usuales en la planificación. Ello representa una modalidad intrínseca que la diferencia cualitativamente de la planificación, al pasar del universo de los números racionales al de los números enteros. Una de las dificultades en los problemas de programación viene marcada por la combinatoria de las soluciones posibles, imposible de explorar, en la mayoría de los casos, de forma exhaustiva.
Otra dificultad añadida es la gran variedad de tipos de problema existentes, tanto por las restricciones a que están sometidos como por los criterios de evaluación utilizados, lo que impide una presentación completa del tema.
De tal forma se divide la programación de operaciones en diversas funciones: (1) La **carga** que se asigna a cada operación en el recurso en que va a ejecutarse; (2) La **secuenciación**, que define el orden o secuencia en que se ejecutarán las operaciones asignadas al mismo recurso; y (3) La **temporización**, que establece las fechas de ejecución.
El uso de buenos programas implica: (a) Desarrollo interrelacionado de las tres funciones, pero por las dificultades que pueden aparecer, normalmente se desarrollan independientemente carga y secuenciación y (b) la temporización es imposible de separar de la secuenciación en muchos casos.
En la programación de operaciones en contexto *JIT* (Just-In-Time) y *DS* (Douki Seisan) encontramos una gran cantidad de problemas combinatorios. Estos problemas tratan en su mayoría sobre utilizar inteligentemente los recursos disponibles para resolver alguno de los aspectos comentados en el ámbito de la programación de operaciones. Uno de los claros ejemplos de gestión eficiente de los recursos disponibles es el caso de las líneas de producción y montaje de automóviles, ya que las unidades que circulan por la línea no son completamente idénticas; algunas de ellas, que se agrupan en familias, poseen cierto grado de similitud, pero pueden variar en diferentes aspectos relacionados con (1) la utilización de recursos en las estaciones de trabajo y (2) el consumo de componentes o partes.
En (Boysen, Fliedner and Scholl, 2009) podemos encontrar una revisión de la literatura sobre los

diferentes problemas que pueden surgir. Este trabajo organiza los principales tipos de problemas en 3 categorías:

1. *Mixed-model sequencing*: Problemas dedicados a completar la mayor cantidad de trabajo posible, dentro de un tiempo disponible y con una cantidad mínima de sobrecarga, usando recursos adicionales si es necesario.

2. *Car sequencing problem*: Problemas dedicados a obtener secuencias que están restringidas en la aplicación consecutiva de opciones especiales (techo solar, acabados especiales,...) en los vehículos circulando en la línea de montaje.

3. *Level scheduling*: Problemas dedicados a obtener secuencias que garanticen la regularidad en el consumo de componentes, conservación del mix de producción en un día laboral y la regularidad en las cargas de trabajo requeridas por los productos.

En el resto de este capítulo vamos a describir brevemente algunos de los problemas que aparecen en este contexto.

1.2. Algunos problemas de secuenciación en el contexto de la programación de operaciones

1.2.1. Mixed-model sequencing problem with workload minimization

Dado que por regla general los productos que se montan en líneas de montaje no son completamente idénticos, pueden existir variaciones en los tiempos de proceso de los productos en las diferentes estaciones. Debido a estas diferencias, y que la línea de montaje normalmente se equilibra para un producto ideal generado a partir del promedio de todos los productos a montar en la línea, es necesario secuenciar los productos adecuadamente para evitar la aparición o reducir la sobrecarga que se puede generar en la línea. Este tipo de líneas de montaje se conoce como líneas de montaje de productos mixtos. Este problema tiene en consideración estas diferencias entre los productos y las configuraciones de las estaciones de la línea, y consiste en obtener una secuencia que minimice la sobrecarga de trabajo de las estaciones que componen la línea de montaje.
Este problema se conoce en la literatura científica como *MMSP-W* (Mixed-model sequencing problem with workload minimization) y fue propuesto por (Yano and Rachamadugu, 1991). Posteriormente, el problema fue tratado por (Scholl, Klein and Domschke, 1998) y (Bautista and Cano, 2008). Recientemente el problema ha sido tratado por (Bautista and Cano, 2011) y (Bautista, Cano and Alfaro, 2012c). El problema se ha tratado de resolver usando diferentes técnicas, como la programación lineal (Bautista, Cano and Alfaro, 2012c), programación dinámica (Yano and Rachamadugu, 1991), metaheurísticas puras e híbridas (Bautista and Cano, 2011) e híperheurísticas (Bautista and Cano, 2008).

1.2.2. Job Shop

El prototipo de problema de secuenciación se encuentra en el denominado Problema del Taller Mecánico (*Job Shop Problem*) cuyo enunciado básico es (Companys (2003)):
"n piezas (lotes, trabajos u órdenes) deben realizarse en m máquinas (recursos, secciones o puestos de trabajo). La realización de cada pieza implica la ejecución, en orden establecido, de una serie de operaciones prefijadas; cada operación está asignada a una máquina concreta y tiene una duración (tiempo de proceso) determinada y conocida. Debe establecerse un programa, es decir, la secuencia de operaciones en cada máquina, que optimice un cierto índice de eficiencia (por ejemplo, la ocupación total del taller)".
Este problema general enmarca diferentes variantes del problema, dependiendo por ejemplo del flujo que han de seguir las piezas en el taller o la existencia o no de *buffers* o espacios de almacenamiento intermedios. Entre las variantes más estudiadas del problema se encuentra el *Flow Shop*, o taller mecánico con flujo regular, en el que todas las piezas tienen la misma ruta: todas piezas pasan por las máquinas en el mismo orden y ninguna pieza recibe más de una operación en cada máquina. Dentro de esta variante, podemos encontrar un subproblema que trata el caso en el que no existen buffers intermedios. A este subproblema se le conoce en la literatura como *BFSP* (Blocking Flow Shop Problem), y se puede encontrar una revisión de la literatura en (Hall and Sriskandarajah, 1996), donde además se demuestra la dificultad del problema cuando el número de máquinas es igual o superior a 3.
En el caso del *BFSP*, este se ha intentado resolver usando algoritmos exactos, como *branch-and-bound* (Levner, 1969), (Companys and Mateo, 2007) y metaheurísticas como por ejemplo *TABU Search* (Grabowski and Pempera, 2007) y algoritmos greedy iterados (Ribas, Companys and Tort-Martorell, 2011).

1.2.3. Output rate variation

En el contexto mencionado anteriormente de las líneas de montaje de productos mixtos, aparecen otras problemáticas vinculadas a las filosofías de trabajo *JIT* y *DS*. Debido a la diversa naturaleza de los productos que en ellas se elaboran, es necesario dotar a éstas de cierta flexibilidad. Dicha flexibilidad condiciona el orden o secuencia en que se han de tratar las unidades en la línea para satisfacer dos principios generales: la reducción drástica del stock de componentes y semielaborados necesarios para ensamblar los productos y emplear de forma eficiente el tiempo disponible para la fabricación.
Alineados con el primer principio surgen una serie de problemas con el objetivo de reducir los stocks de los componentes necesarios para los productos alrededor de la línea. Es en esta situación donde aparece el problema *ORV* (Output rate variation), que tiene como objetivo mantener lo más regular posible el consumo de componentes a lo largo del proceso productivo. Dicho objetivo esta alineado con el objetivo de reducir los stocks intermedios y, por tanto, vinculado con la

filosofía de trabajo *JIT*.
Este problema fue propuesto de forma pionera por (Monden, 1998), que fue publicado por primera vez en 1983, dedicado al sistema productivo de la empresa automovilística Toyota y su denominación se debe a (Kubiak, 1993).
Para resolver este problema se han propuesto diferentes heurísticas como el algoritmo constructivo *goalchasing* (Monden, 1998), algoritmos de hormigas (Bautista, Pereira and Cano, 2010) y también algunos algoritmos exactos como la programación dinámica acotada (Bautista, 1993), (Bautista, Companys and Corominas, 1996a).

1.2.4. Product rate variation

En el mismo contexto del problema anterior, también es deseable mantener la salida de productos de la línea lo más regular posible. Es en el marco del problema anterior *ORV* donde surge el problema *PRV* (Product rate variation). Este problema, cuyo objetivo es regularizar la salida de productos de la línea de montaje mixta, se propone en (Miltenburg, 1989), donde se simplifica el problema *ORV* en algunos casos. Posteriormente, en (Kubiak, 1993) se le da nombre por primera vez. El problema *PRV* puede ser considerado como un caso particular del *ORV* (Bautista, 1993).
Para resolver este problema se han propuesto procedimientos exactos, como la programación lineal (Bautista, Companys and Corominas, 1997) y se ha propuesto la resolución a través de algoritmos para resolver problemas de asignación (Bautista et al., 1997).

1.2.5. Car-sequencing problem

El *CSP* (Car-Sequencing Problem) es un problema en el marco de la secuenciación de unidades en líneas de montaje mixtas. El problema, propuesto por (Parello, Kabat and Wos, 1986), consiste en establecer una secuencia de coches (o productos) de diferentes tipos. Cada uno de los productos requiere un conjunto de opciones, que normalmente requieren un trabajo en determinadas estaciones superior al tiempo estándar, establecido en consonancia con el ratio de producción requerido a la línea. A consecuencia de este sobreesfuerzo, la secuencia de producción requiere el cumplimiento de una serie de restricciones sobre el orden en que los productos circulan en la línea, para satisfacer el número consecutivo de unidades que pueden llevar una determinada opción (por ejemplo sólo 2 de cada 5 coches pueden llevar techo solar).
Para resolver este procedimiento se han propuesto diferentes heurísticas como *GRASP* y *Beam Search* (Bautista, Pereira and Adenso-Díaz, 2008), algoritmos basados en búsqueda local (Putcha and Gottlieb, 2002) o algoritmos de hormigas (Gottlieb, Puchta and Solnon, 2003). También se ha intentado resolver a través de procedimientos exactos como la programación lineal (de hecho el problema *CSP* es utilizado por muchos paquetes informáticos para la resolución de programas lineales como uno de los ejemplos de uso).

1.3. Algunos procedimientos de resolución

En este apartado se describe brevemente algunos de los procedimientos típicos para la resolución de problemas en el contexto de la programación de operaciones.

1.3.1. Procedimientos exactos

Programación lineal

La Programación Lineal es un procedimiento vinculado a algoritmos matemáticos usado en la resolución de problemas. El problema se formula a través de ecuaciones lineales y consiste en optimizar una función objetivo, también lineal.
Dicho procedimiento de optimización consiste en encontrar el mejor valor (ya sea en sentido de minimización o de maximización) para una función objetiva, conocida como función objetivo, de tal forma que las variables que forman dicha función están sujetas a una serie de restricciones que se formulan a través de un sistema de inecuaciones lineales.

Programación dinámica

La Programación dinámica en el contexto de la informática puede ser utilizada de diversas maneras. La primera de ellas puede ser para reducir el tiempo de ejecución de un algoritmo mediante la utilización de subproblemas. La segunda se basa en la descripción de los estados posibles de un problema y las posibles transiciones entre los estados. El matemático Richard Bellman inventó la programación dinámica en 1953 que se utiliza para optimizar problemas complejos. Cuando hablamos de programación dinámica acotada (*BDP*), nos referimos a un procedimiento que incorpora características de la programación dinámica (determinación de caminos extremos en grafos) con características de los algoritmos *branch-and-bound*. Los principios de la programación dinámica acotada fueron descritos por (Bautista, 1993) y (Bautista et al., 1996a). Trabajos previos en heurísticas similares fue realizado por (Morin and Marsten, 1976; Marsten and Morin, 1978) y extendido por (Carraway and Schmidt, 1991).

1.3.2. Procedimientos Metaheurísticos

Antes de nada, definamos el termino metaheurística, acuñado por primera vez en el artículo seminal sobre búsqueda tabú de Fred Glover en 1986:
"Las metaheurísticas, en su definición original, son métodos de resolución que orquestan una interacción entre procedimientos de mejora local y estrategias de alto nivel para crear procesos capaces de escapar de óptimos locales y realizar una búsqueda robusta de un espacio de soluciones. A lo largo del tiempo, estos métodos han incluido procedimientos que emplean estrategias que superan la trampa de los óptimos locales en espacios de soluciones complejos, especialmente en aquellos procedimientos que utilizan una o más

estructuras de vecindarios como medio para definir movimientos de transición admisibles de una solución hacia otra, o para crear o destruir soluciones en procesos constructivos o destructivos"[...]
[...] *´´Aunque las metaheurísticas no son capaces de certificar la optimalidad de las soluciones que encuentran, los procedimientos exactos se han demostrado incapaces de encontrar soluciones cuya calidad se aproxime a las obtenidas a través de metaheurísticas - particularmente en los problemas reales, que a menudo alcanzan altos niveles de complejidad."*

Traducción del inglés, Fuente: (Glover and Kochenberger, 2003), prefacio.

Una posible clasificación en un marco general de las metaheurísticas es:

- Las metaheurísticas constructivas se orientan a los procedimientos que tratan de la obtención de una solución a partir del análisis y selección paulatina de las componentes que la forman.
- Las metaheurísticas de búsqueda guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explotar las estructuras de entornos asociadas.
- Las metaheurísticas evolutivas están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

A continuación podemos ver con más detalles cada una de las categorías, con algunos ejemplos de esquemas metaheurísticos. Posteriormente hablaremos de la posibilidad de hibridación de los diferentes esquemas.

Metaheurísticas constructivas

Las heurísticas constructivas aportan soluciones del problema por medio de un procedimiento que incorpora iterativamente elementos a una estructura, inicialmente vacía, que representa a la solución. Las metaheurísticas constructivas utilizan estrategias para seleccionar las componentes con las que se construye una buena solución del problema, pero sin replantearse las decisiones ya tomadas.
Entre las metaheurísticas que pertenecen a esta categoría se encuentra la estrategia voraz o *greedy*, que implica la elección que da mejores resultados inmediatos, sin tener en cuenta una perspectiva mas amplia. Dentro de este tipo de metaheurística, destaca la aportación de la metaheurística *greedy randomized adaptive search procedure* (*GRASP*) que, en la primera de sus dos fases, incorpora a la estrategia *greedy* pasos aleatorios con criterios adaptativos para la selección de los elementos a incluir en la solución.

- Algoritmos voraces. A este tipo de algoritmos voraces (también conocido como *greedy*) corresponden aquellos que para resolver un determinado problema, siguen una heurística consistente en elegir la mejor opción de entre todos los elementos candidatos en cada paso con

la esperanza de llegar a una buena solución. Dada la facilidad de plantear este tipo de algoritmos, es un esquema muy popular. En este esquema en ningún caso se replantean las decisiones tomadas.

- *GRASP*. El nombre *GRASP* (del inglés *greedy randomized adaptive search procedure*) es un método de multi-inicio iterativo en donde cada iteración tiene dos fases: (i) construcción y (ii) búsqueda local. En la fase de construcción se encuentra una solución factible y posteriormente su vecindad es explorada a través de una búsqueda local hasta llegar a un mínimo local. Esta solución se guarda y se suele empezar la primera fase, ya que en esta se suele incluir azar para diversificar las soluciones que se generan.

Metaheurísticas de búsqueda

Otro tipo de metaheurísticas son las de búsqueda. Este tipo de metaheurísticas establecen estrategias para recorrer el espacio de soluciones del problema transformando de forma iterativa soluciones de partida. Las búsquedas evolutivas se distinguen de éstas en que es un conjunto de soluciones, generalmente llamado población, el que evoluciona sobre el espacio de búsqueda.
Este tipo de heurística normalmente aplican alguna regla de forma inteligente para mejorar la solución de un problema. Dicha regla se aplica de forma iterativa mientras sea posible obtener nuevas mejoras. Dentro de esta categoría de metaheurísticas existen diferentes estrategias de búsqueda, como las búsquedas monótonas, algoritmos escaladores (*hill-climbing*) o búsquedas locales. Esta última denominación obedece a que la mejora se obtiene en base al análisis de soluciones similares a la que realiza la búsqueda, denominadas soluciones vecinas. Entre las más populares se encuentran la búsqueda tabú, los procedimientos multi-arranque, las búsquedas de entorno variable o el recocido simulado.

- Búsqueda Tabú. La búsqueda tabú pertenece a un tipo de búsquedas que utilizan la memoria del ordenador para guardar información sobre el recorrido ya realizado para evitar que la búsqueda se concentre en una misma zona del espacio.
  Con ese fin, se crean listas en memoria que prohíben temporalmente soluciones muy parecidas a las últimas soluciones del recorrido, con el fin de escapar de óptimos locales. Es por el uso de estas listas, que reciben el nombre de listas tabú, por lo que se nombra a este tipo de búsquedas como búsquedas tabú.

- Búsquedas Multi-Arranque. Las búsquedas Multi-arranque proponen un esquema en el que se alternan una fase de generación de soluciones con otra de mejora de las mismas. El proceso se vuelve a repetir hasta que se cumple un determinando criterio de finalización.
  Las búsquedas multi-arranque más básicas que podemos considerar es aquella en la que las soluciones iniciales se generan al azar, y la fase de búsqueda se realiza mediante algún procedimiento de búsqueda local desde cada uno de los puntos generados.

- Búsqueda en Entorno variable. La búsqueda por Entornos variables (del inglés *Variable Neighbourhood Search*, *VNS*) se basa en un principio simple: cambiar sistemáticamente la estructura de los entornos por los que se realiza la búsqueda.
  Con ese fin, se va variando los operadores y estrategias de búsqueda, ya sea de forma determinística o al azar, para diversificar el espacio de búsqueda con el objetivo de evitar los óptimos locales.

- Recocido simulado. Las búsquedas basadas en el recocido simulado (del inglés *Simulated annealing*, *SA*) se basan en encontrar una buena aproximación al valor óptimo de una función. Este tipo de búsquedas, empleadas por primera vez en 1983, se basan en la observación del proceso de recocido del acero.
  Este esquema se basa en la posibilidad de permitir empeorar las soluciones con una cierta probabilidad en algunos momentos de la búsqueda, de nuevo con el fin de escapar de óptimos locales.

Metaheurísticas evolutivas

Las metaheurísticas evolutivas establecen estrategias para conducir la evolución en el espacio de búsqueda de conjuntos de soluciones (usualmente llamados poblaciones) con la intención de acercarse a la solución óptima con los elementos que la componen.
El aspecto fundamental que marca la diferencia entre las metaheurísticas evolutivas y las de búsqueda consiste en que las de tipo evolutivo la interacción se realiza entre los miembros de la población frente, mientras que en las de búsqueda se guían por la información de soluciones individuales.
Las diferentes metaheurísticas evolutivas se distinguen por la forma en que combinan la información proporcionada por los elementos de la población para hacerla evolucionar mediante la obtención de nuevas soluciones.
La principal diferencia entre los diferentes tipos de metaheurísticas evolutivas es el uso de procedimientos aleatorios o determinísticos para el cruce de poblaciones. El algunos casos se usa el azar, como es el caso de los algoritmos genéticos, mientras que por ejemplo en la búsqueda dispersa se emplea procedimientos deterministas.

- Búsqueda dispersa. La Búsqueda Dispersa (del inglés *Scatter Search*) es una metaheurística evolutiva propuesta a comienzo de la década de los setenta. Este esquema se basa en estrategias para combinar reglas de decisión, especialmente en problemas de secuenciación, así como en la combinación de restricciones.
  La Búsqueda dispersa se basa en el principio de que dadas dos soluciones, se puede obtener una nueva mediante su combinación de modo que mejore a las que la originaron. Usualmente la búsqueda dispersa se combina con un procedimiento denominado *Path Relinking*.

- Algoritmos de hormigas. Los algoritmos de hormigas están inspirados en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento. En dicho comportamiento, las hormigas vagan aleatoriamente en su búsqueda de alimento, y a lo largo de su camino de regreso a la colonia depositan una hormona denominada feromona. Si otras hormigas encuentran este rastro, lo más probable es que sigan este camino para depositar el alimento en la colonia. Con el paso del tiempo el rastro de la feromona comienza a evaporarse y se reduce su fuerza atractiva. Las hormigas que siguen el rastro aumentan la cantidad de la feromona, con lo que el rastro es más fuerte y dura más tiempo. Cuantas más hormigas recorran ese camino, más intenso será el olor de la feromona, lo que estimula a más hormigas a seguir esa trayectoria.
  Desde el punto de vista algorítmico, una población inicial de hormigas generan soluciones. En función de la calidad de la solución construida se deposita la feromona. Posteriormente, se evaporan los rastros de feromona para diversificar las soluciones y se vuelve a empezar el proceso. Se puede incorporar un paso de búsqueda local para intentar mejorar la calidad de las soluciones.

- Algoritmos genéticos. Los algoritmos genéticos nacen en el año 1970. Son llamados de esta forma debido a que se inspiran en la evolución biológica.
  Estos algoritmos hacen evolucionar una población de individuos inicial, sometiendo a la población a acciones al azar similares a la evolución biológica (mutaciones y recombinaciones genéticas). Posteriormente, los individuos son escogidos con algún criterio, decidiéndose así cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

Metaheurísticas híbridas

Aunque lo que se ha descrito es un posible clasificación de las metaheurísticas, cabe destacar que muchas de ellas no son puras, ya que como se ha visto algunas de ellas por ejemplo, utilizan parte de una metaheurística constructiva y parte de una de búsqueda.
A este tipo de metaheurísticas, que utilizan propiedades de más de un esquema metaheurístico se les denomina metaheurísticas híbridas.

1.4. La programación dinámica acotada

Los procedimientos que vamos a proponer en esta tesis para resolver los diferentes problemas están basados principalmente en la programación dinámica acotada.
La Programación dinámica acotada (del inglés *Bounded Dynamic Programming, BDP)* tiene como origen la programación dinámica y los procedimientos *Branch-and-bound*.

La programación dinámica es una técnica de resolución de problemas combinatorios que consiste en definir un grafo asociado al problema y encontrar sobre dicho grafo un camino extremo. En dicho grafo los vértices corresponden a los estados y los arcos a las decisiones. Este método tiene su origen en (Bellman, 1954).
Por otra parte, los procedimientos *Branch-and-bound* son algoritmos exactos para la resolución de problemas, en los que se usa conocimiento sobre el problema, a través de cotas inferiores, para reducir el espacio de búsqueda.
En (Bautista, 1993) encontramos una descripción detallada del procedimiento *BDP*, aplicada al procedimiento *ORV*. Sin embargo, los orígenes de la *BDP* se remontan al trabajo de (Ibaraki, 1987), donde por primera vez se relacionan los procedimientos basados en programación dinámica y los procedimientos *Branch-and-bound*. Anteriormente en (Morin and Marsten, 1976), y (Marsten and Morin, 1978) se aplica el uso de cotas a la programación dinámica. (Carraway and Schmidt, 1991) recogen la propuesta de los anteriores autores y la extienden para la resolución de otros problemas.
El procedimiento *BDP* consiste en la utilización de cotas en un esquema de programación dinámica. Además el procedimiento incorpora diversos tratamientos para limitar el número de vértices del grafo que se almacena en memoria. Para implementar el procedimiento *BDP* en este caso se utilizará el lenguaje de programación C++.

1.5. Hipótesis de trabajo

Las hipótesis de trabajo de la investigación son las siguientes:

- Las líneas de montaje en entornos con ideologías *JIT* o *DS*, deben ser capaces de tratar diferentes tipos de productos. Lo que hace que la flexibilidad sea una característica deseable en este tipo de líneas.

- La flexibilidad de las líneas hace necesaria la secuenciación de las unidades de producto, con el fin de maximizar su eficiencia, y por tanto, evitar sobrecargas de trabajo o tiempos muertos, mantener el consumo regular de componentes y favorecer la finalización de la producción en la planta lo antes posible.

- Una secuencia regular de productos, favorece la no concentración de sobrecarga en un momento determinado o una estación de trabajo concreta. Además de favorecer la regularidad del consumo de componentes, idea básica dentro de la ideología *JIT*.

- El procedimiento *BDP* ofrece buenos resultados cuando los problemas disponen de buenas cotas y es posible definir el grafo de estados.

- El procedimiento *BDP* es competitivo en tiempo de resolución, no solamente en resultados ofrecidos.
- En caso de no disponer de buenas cotas, el uso de la programación lineal puede ayudar a establecer cotas para el problema.
- En el caso de no disponer soluciones para las instancias usadas, la programación lineal ofrecerá resultados de partida en un tiempo razonable.

1.6. Objetivos y resultados esperados

El objetivo central de esta tesis es estudiar el comportamiento de la programación dinámica acotada en los problemas *MMSP-W* (en diferentes variantes), *BFSP* y *ORV*.

1.6.1. Sobre el *MMSP-W*

Modelos para el problema

Partiendo de los modelos de referencia de la literatura de (Yano and Rachamadugu, 1991) y (Scholl et al., 1998), se formularán nuevos modelos teniendo en cuenta la consideración de vínculos entre estaciones en serie, la libre interrupción (o condicionada) de las operaciones en las estaciones, y finalmente la incorporación de la regularidad en el mix de producción.

Algoritmo basado en programación dinámica acotada *BDP*

Con el fin de implementar un procedimiento basado en *BDP* (programación dinámica acotada o bounded dynamic programming), se desarrollaran las cotas y mecanismos necesarios para utilizar el procedimiento para resolver las diferentes variantes tomadas en consideración. Esto incluye el desarrollo del grafo asociado al problema, cotas y dominancias que pudieran ser necesarias. Este punto incluye la programación en C++ del procedimiento.

Experiencia computacional

Dado que en la literatura no existen ejemplares de referencia públicos, el presente trabajo propondrá unos ejemplares de acceso público. Estos ejemplares consistirán en dos conjuntos de ejemplares: uno basado en un juego de pruebas y otro basado en un caso de estudio vinculado a la línea de producción de motores de Nissan Spanish Industrial Operations (*NSIO*). Dado que no se conocen los resultados de estos ejemplares, con el fin de constatar la validez del procedimiento, se resolverán mediante programación lineal entera mixta, utilizando herramientas de resolución como Gurobi o Cplex.

Posteriormente se resolverán los ejemplares usando el procedimiento *BDP* y se analizarán los resultados.

Objetivos

El objetivo de este trabajo de investigación será el de implementar procedimientos competitivos (tanto en tiempo de proceso como en calidad de resultados) basados en *BDP* para resolver las variantes tomadas en consideración, así como implementar otros procedimientos para contrastar la calidad del procedimiento propuesto.

1.6.2. Sobre el *BFSP*

En el problema *BFSP* estudiaremos el problema original presente en la literatura, así como variantes asociadas a la incorporación de la regularidad en el mix de producción. Los pasos a desarrollar son los que se describen a continuación.

Algoritmo *BDP*

Con el fin de implementar el procedimiento basado en *BDP*, se desarrollaran las cotas y mecanismos necesarios para utilizar el procedimiento para resolver el problema *BFSP*. Esto incluye el desarrollo del grafo asociado al problema, cotas y dominancias que pudieran ser necesarias. Este punto incluye la programación en C++ del procedimiento.

Experiencia computacional

Dado que en la literatura en este caso si existen ejemplares de referencia públicos (Taillard, 1993) y que la mejor solución de estos se encuentra recogida en (Ribas et al., 2011), en este caso se resolverán dichos ejemplares usando el procedimiento *BDP*. Posteriormente, se analizarán los resultados.

Objetivos

De nuevo, el objetivo de este trabajo de investigación será el de implementar procedimientos competitivos (tanto en tiempo de proceso como en calidad de resultados) basados en *BDP* para resolver el problema *BFSP*.

1.6.3. Sobre el *ORV*

En el problema *ORV* estudiaremos el problema original presente en la literatura, así como variantes asociadas a la incorporación de la regularidad en el mix de producción. También se estudiará la introducción de una restricción adicional para acotar el espacio de búsqueda.

Algoritmo *BDP*

Con el fin de implementar el procedimiento basado en *BDP*, se desarrollaran las cotas y mecanismos necesarios para utilizar el procedimiento para resolver el problema *ORV*. Esto incluye el desarrollo del grafo asociado al problema, cotas y dominancias que pudieran ser necesarias. Este punto incluye la programación en C++ del procedimiento.

Experiencia computacional

Dado que en la literatura en este caso sí existen ejemplares de referencia públicos ((Bautista et al., 1996a); (Jin and Wu, 2002)) y que la mejor solución de estos se encuentra recogida en la literatura, se resolverán dichos ejemplares usando el procedimiento *BDP*. Posteriormente, se analizarán los resultados.

Objetivos

De nuevo, el objetivo de este trabajo de investigación será el de implementar procedimientos competitivos (tanto en tiempo de proceso como en calidad de resultados) basados en *BDP* para resolver el problema *ORV*.

Capítulo 2

El *MMSP-W*

2.1. Introducción y estado del arte

Como se ha comentado previamente, este problema aparece en líneas de montaje mixtas donde se producen diferentes tipos de productos, similares entre sí.
En el *MMSP-W*, un número $|I|$ de tipos de producto han de ser secuenciados para que sean procesados en un conjunto de K máquinas. Cada tipo de producto tiene establecido una demanda d_i que debe ser producida. Además, cada unidad de tipo de producto requiere a cada estación un tipo de proceso determinado, en cada una de las estaciones $k \in K$, y a cada una de las estaciones se le concede un tiempo de trabajo durante el cual puede trabajar sobre una unidad de producto. A este tiempo se le conoce como tiempo de ciclo (c). En algunas ocasiones, con el fin de permitir procesar las unidades que requieren un tiempo de trabajo superior al del ciclo y completar el trabajo sobre dichas unidades, la estación puede retener dicha unidad un tiempo superior al del tiempo de ciclo, conocido como tiempo de ventana o ventana temporal (l_k). En el caso de retener la unidad, esto afecta en la reducción del tiempo disponible para procesar la siguiente unidad de producto. Cuando alcanzamos la ventana temporal, es obligatorio liberar la unidad para la siguiente estación, produciéndose en ese caso el efecto que se conoce como sobrecarga.
El objetivo del problema es encontrar una secuencia de producción de las diferentes unidades de productos que minimice la sobrecarga global del sistema, o lo que es lo mismo, que maximice el trabajo total completado.
La sobrecarga se mide en unidades de tiempo y se corresponde con el trabajo que no se ha podido completar en la línea de montaje. Ante la aparición de dicho fenómeno se pueden tomar diferentes medidas, tal y como aparece en la literatura:

1. Parar la línea con el fin de completar el trabajo pendiente (Okamura and Yamashina, 1979; Xiaobo and Ohno, 1997).

2. Dejar pasar la unidad a la siguiente estación, dejando el trabajo no completado para otro

momento (Yano and Rachamadugu, 1991; Scholl et al., 1998; Bautista and Cano, 2008).

3. Aumentar la capacidad de la línea mediante operarios refuerzo (Boysen and Bock, 2011).

En esta tesis trataremos el problema *MMSP-W*, debido a la complejidad del mismo. Para este trabajo partiremos de los modelos de referencia de (Yano and Rachamadugu, 1991) que tiene como objetivo maximizar el trabajo completado en la línea y el modelo de (Scholl et al., 1998) con el objetivo de minimizar la sobrecarga de trabajo. Además, el problema *MMSP-W* admite diferentes variantes como la libre interrupción de tareas o la interrupción condicionada, sólo permitida en algunos instantes. Para su resolución emplearemos los procedimientos basados en *BDP* así como otros procedimientos metaheurísticos.
Parte del trabajo presentado en este capítulo se puede encontrar en (Bautista and Cano, 2011; Bautista, Cano and Alfaro, 2012a; Bautista, Cano and Alfaro, 2012b; Bautista, Cano and Alfaro, 2012c; Bautista, Cano and Alfaro, 2012d; Bautista, Alfaro and Cano, 2013).

2.1.1. Estado del arte

En (Boysen et al., 2009), encontramos una revisión de la literatura sobre el *MMSP-W*. Este problema es *NP-hard*, demostrado por (Yano and Rachamadugu, 1991).
Debido al hecho de que el problema es *NP-hard*, en la literatura se encuentran varios procedimientos heurísticos para el problema *MMSP-W*. A continuación se recogen algunos ejemplos de las heurísticas desarrolladas en la literatura, para algunas de las variantes del *MMSP-W*:

- (Yano and Bolat, 1989) presentaron un procedimiento heurístico basado en búsqueda local.
- (Yano and Rachamadugu, 1991) propusieron un procedimiento basado en programación dinámica.
- (Bolat and Yano, 1992) propusieron un procedimiento greedy basado en reglas de prioridad.
- (Okamura and Yamashina, 1979) y (Scholl et al., 1998) presentaron procedimientos basados en una metaheurística para resolver algunas de las variantes del problema.
- (Bolat, 2003) propuso un algoritmo exactos basados en branch and bound.
- (Bautista and Cano, 2008) propusieron un procedimiento basado en híper heurísticas y el uso de reglas de prioridad.
- (Cano, Ríos and Bautista, 2010) presentaron un algoritmo basado en búsqueda dispersa.
- (Erel, Gocgun and Sabuncuoglu, 2007) propusieron un algoritmo basado en Beam Search.

También existen precedentes de problemas multi-criterio relacionados de una forma u otra con el *MMSP-W* (Aigbedo and Monden, 1997; Ding, Zhu and Sun, 2006; Kotani, Ito and Ohno, 2004; Rahimi-Vahed and Mirzaei, 2007).

2.2. Modelos de referencia

Para el problema de secuenciación de productos mixtos en una línea de producción con múltiples estaciones, con el propósito de minimizar la sobrecarga de trabajo, partiremos de dos modelos recogidos en la literatura y que son considerados como referencia:

- $M1$: Modelo propuesto por (Yano and Rachamadugu, 1991) que está orientado a maximizar el trabajo total completado en la línea.

- $M2$: Modelo propuesto por (Scholl et al., 1998) cuyo objetivo es minimizar la sobrecarga de trabajo o esfuerzo adicional.

Los modelo $M1$ y $M2$ son válidos cuando no se consideran vínculos entre estaciones dispuestas en serie en la línea, de forma que el instante de inicio de una operación, sobre una unidad de producto, en una estación concreta, no depende del instante de finalización de la operación realizada en la estación inmediatamente anterior sobre esa misma unidad de producto.
El modelo $M1$ tiene por objeto maximizar el valor del trabajo completado. Seguidamente, definimos sus parámetros y variables:

Parámetros

K	Conjunto de estaciones de trabajo ($k = 1, ..., \|K\|$).
b_k	Número de procesadores homogéneos en cada estación de trabajo ($k = 1, ..., \|K\|$).
I	Conjunto de tipos de producto ($i = 1, ..., \|I\|$).
d_i	Demanda programada del tipo de producto i.
$p_{i,k}$	Tiempo de proceso requerido a cada procesador homogéneo (*en actividad normal*), por una unidad de producto de tipo i en la estación k.
T	Demanda total. Obviamente $\sum_{i=1}^{\|I\|} d_i = T$.
t	Indice de posición en la secuencia $t = 1, ..., T$.
c	Tiempo de ciclo. Tiempo estándar asignado a cada procesador de las estaciones de trabajo, $k = 1, ..., \|K\|$, para tratar cualquier unidad de producto.
l_k	Ventana temporal. Tiempo máximo que se le permite, a cada procesador de la estación k, trabajar en cualquier unidad de producto; siendo $l_k - c > 0$ el tiempo máximo que se puede retener una unidad de producto, en la estación k, una vez concluido el ciclo.

Variables

$x_{i,t}$ Variable binaria que adopta el valor 1 si una unidad del producto i ($i = 1, ..., |I|$) se asigna a la posición t ($t = 1, ..., T$) de la secuencia, y valor 0 en caso contrario.

$s_{k,t}$ Instante de inicio del trabajo aplicado a la *t-ésima* unidad de la secuencia de productos en la estación k ($k = 1, ..., |K|$).

$v_{k,t}$ Tiempo de proceso aplicado, por cada procesador homogéneo (*en actividad normal)* a la *t-ésima* unidad de producto secuenciada en la estación de trabajo k.

En tales condiciones, (Yano and Rachamadugu, 1991) formularon el siguiente programa matemático, $M1$:

$$Max \quad V = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} v_{k,t} \right) \tag{2.1}$$

Sujeto a:

$$\sum_{t=1}^{T} x_{i,t} = d_i \qquad i = 1, ..., |I| \tag{2.2}$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad t = 1, ..., T \tag{2.3}$$

$$v_{k,t} \leqslant \sum_{i=1}^{|I|} p_{i,k} x_{i,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.4}$$

$$s_{k,t} \geqslant (t-1)c \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.5}$$

$$s_{k,t} \geqslant s_{k,t-1} + v_{k,t-1} \qquad k = 1, ..., |K|; t = 2, ..., T \tag{2.6}$$

$$s_{k,t} + v_{k,t} \leqslant (t-1)c + l_k \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.7}$$

$$v_{k,t} \geqslant 0; s_{k,t} \geqslant 0 \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.8}$$

$$x_{i,t} \in \{0, 1\} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.9}$$

Tomando como referencia temporal de inicio de las operaciones $s_{k,1} = 0, \forall k$, podemos considerar que, en $M1$, los instantes mínimos de inicio de las operaciones son $s_{k,t}^{min} = (t-1)c, \forall k, \forall t$.
En el modelo, V representa el valor del trabajo total realizado; las restricciones (2.2) obligan a satisfacer la demanda programada; las restricciones (2.3) indican que en cada posición de la secuencia sólo se puede asignar una unidad de producto; las restricciones (2.4) establecen que el tiempo de proceso aplicado a cada unidad en cada estación esta limitado por su tiempo de proceso requerido; las restricciones (2.5), (2.6) y (2.7) establecen los límites para los instantes de inicio de las operaciones y los tiempos de proceso aplicados a las unidades; las restricciones (2.8) indican que los tiempos de proceso aplicados y los instantes de inicio son no negativos; y, finalmente, las restricciones (2.9) fuerzan la binariedad de las variables de asignación.

Nótese que los coeficientes b_k representan el número de procesadores en cada estación, en lugar del número de operarios como en (Yano and Rachamadugu, 1991), para poder referirnos, indistintamente a personas, robots o sistemas persona-máquina.

El modelo $M2$ tiene por objeto minimizar la sobrecarga total, W. Seguidamente, definimos las variables adicionales:

Variables adicionales

$\hat{s}_{k,t}$ Diferencia positiva entre el instante de inicio e instante mínimo de inicio de la *t-ésima* operación en la estación k. $\hat{s}_{k,t} = [s_{k,t} - (t-1)c]^+$ (con $[x]^+ = max\{0, x\}$).

$w_{k,t}$ Sobrecarga generada, medida en tiempo, por la *t-ésima* unidad de la secuencia de productos en la estación k.

$\rho_{k,t}$ Tiempo de proceso requerido por la *t-ésima* unidad de la secuencia de productos en la estación k.

En tales condiciones, (Scholl et al., 1998) formularon el siguiente modelo, $M2$:

$$Min \quad W = \sum_{k=1}^{|K|} \sum_{t=1}^{T} w_{k,t} \tag{2.10}$$

Sujeto a:

$$\sum_{t=1}^{T} x_{i,t} = d_i \qquad i = 1, ..., |I| \tag{2.11}$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad t = 1, ..., T \tag{2.12}$$

$$\rho_{k,t} = \sum_{i=1}^{|I|} p_{i,k} x_{i,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.13}$$

$$\hat{s}_{k,t+1} \geqslant \hat{s}_{k,t} + \rho_{k,t} - w_{k,t} - c \qquad k = 1, ..., |K|; t = 1, ..., T-1 \tag{2.14}$$

$$\hat{s}_{k,t} + \rho_{k,t} - w_{k,t} \leqslant l_k \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.15}$$

$$\hat{s}_{k,t} \geqslant 0; w_{k,t} \geqslant 0 \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.16}$$

$$\hat{s}_{k,1} = 0; \hat{s}_{k,T+1} = 0 \qquad k = 1, ..., |K| \tag{2.17}$$

$$x_{i,t} \in \{0, 1\} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.18}$$

En el modelo, la función objetivo (2.10) representa la minimización de la sobrecarga total de trabajo; las ecuaciones (2.13) vinculan los tiempos de proceso requeridos por los tipos de producto con los tiempos de proceso requeridos por las unidades en la secuencia; finalmente, las restricciones (2.14), (2.15), (2.16) y (2.17) establecen los límites inferior y superior de los instantes relativos de inicio de las operaciones, y su vinculación con los tiempos de proceso requeridos y las sobrecargas.

2.3. *MMSP-W* con estaciones en serie considerando instantes de interrupción restringidos

Como se ha comentado anteriormente, los modelo $M1$ y $M2$ son válidos cuando no se consideran vínculos entre estaciones dispuestas en serie en la línea, de forma que el instante de inicio de una operación, sobre una unidad de producto, en una estación concreta, no depende del instante de finalización de la operación realizada en la estación inmediatamente anterior sobre esa misma unidad de producto.
Por ello, en esta sección proponemos dos modelos, $M3$ y $M4$, que pueden verse como modelos extendidos de $M1$ y $M2$, respectivamente y que sí tienen en consideración los vínculos que existen, a través del *Work In Progress*, entre estaciones de trabajo vecinas de una línea de producción con estaciones dispuestas en serie. Estos modelos servirán como base para los modelos que contengan instantes de finalización condicionados.

2.3.1. Modelos de referencia con vínculos entre estaciones

Los modelos $M1$ y $M2$ no pueden aplicarse a líneas de producción con vínculos entre estaciones, debido a que, en ambos modelos, los instantes de inicio en una estación solamente dependen del ciclo y el instante de finalización (o interrupción) de la operación anterior en la misma estación, cuando la estación es liberada.
Los vínculos entre estaciones consecutivas pueden condicionar los instantes de inicio de las operaciones en las siguientes situaciones:

1. Cuando el proceso que se puede aplicar a una unidad de producto en una estación determinada requiere completar el proceso en la estación previa, excepto en la primera estación.

2. Cuando una unidad de producto esta bloqueando una estación debido a que la siguiente estación está ocupada y no existen espacios de almacenamiento entre estaciones.

Debido a estos motivos, es necesario extender los modelos de referencia propuestos por (Yano and Rachamadugu, 1991) y (Scholl et al., 1998) para tener en cuenta dichos vínculos. El primero de los modelos propuestos, el modelo $M3$, tiene por objeto maximizar el valor del trabajo completado y puede entenderse como una extensión del modelo de (Yano and Rachamadugu, 1991), teniendo en cuenta varios procesadores y vínculos entre estaciones. Sus parámetros y variables son los mismos utilizados por el modelo $M1$. A continuación se encuentra el modelo matemático $M3$:

$$Max \quad V = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} v_{k,t} \right) \tag{2.19}$$

Sujeto a:

$$\sum_{t=1}^{T} x_{i,t} = d_i \qquad i = 1, ..., |I| \tag{2.20}$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad t = 1, ..., T \tag{2.21}$$

$$v_{k,t} \leqslant \sum_{i=1}^{|I|} p_{i,k} x_{i,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.22}$$

$$s_{k,t} \geqslant (t + k - 2)c \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.23}$$

$$s_{k,t} \geqslant s_{k,t-1} + v_{k,t-1} \qquad k = 1, ..., |K|; t = 2, ..., T \tag{2.24}$$

$$s_{k,t} \geqslant s_{k-1,t} + v_{k-1,t} \qquad k = 2, ..., |K|; t = 1, ..., T \tag{2.25}$$

$$s_{k,t} + v_{k,t} \leqslant (t - k - 2)c + l_k \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.26}$$

$$v_{k,t} \geqslant 0; s_{k,t} \geqslant 0 \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.27}$$

$$x_{i,t} \in \{0, 1\} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.28}$$

Tomando como referencia temporal de inicio de las operaciones $s_{1,1} = 0$, podemos considerar que, en $M3$, los instantes mínimos de inicio de las operaciones son $s_{k,t}^{min} = (t + k - 2)c; \forall k, \forall t$.
Esta hipótesis es valida cuando las unidades del producto (por ejemplo motores o coches) se mueven por la línea de montaje a velocidad constante y la estación debe esperar a que el producto llegue a su posición antes de comenzar sus operaciones.
En el modelo, V representa el valor del trabajo total realizado; las restricciones (2.20) obligan a satisfacer la demanda programada; las restricciones (2.21) indican que en cada posición de la secuencia sólo se puede asignar una unidad de producto; las restricciones (2.22) establecen que el tiempo de proceso aplicado a cada unidad en cada estación esta limitado por su tiempo de proceso requerido; las restricciones (2.23), (2.24) y (2.25) establecen que el instante de inicio mas temprano para la *t-ésima* unidad de producto de la secuencia en la estación k es el máximo de entre los siguientes: el tiempo de llegada esperado del producto al principio de la estación ($s_{k,t}^{min}$), el instante en que dicha estación finaliza sus operaciones en el producto previo ($t-1$, o el instante en el cual la estación previa ($k-1$) libera la *t-ésima* unidad; las restricciones (2.26) limitan el máximo tiempo de proceso que se puede aplicar a la *t-ésima* unidad de la secuencia en la estación k; Las restricciones (2.27) indican que los tiempos de proceso aplicados y los instantes de inicio son no negativos; y, finalmente, las restricciones (2.28) fuerzan la binariedad de las variables de asignación.
El segundo modelo de referencia propuestos, el modelo $M4$, tiene por objeto minimizar la sobrecarga total W y puede verse como una extensión del modelo propuesto por (Scholl et al., 1998), considerando además la posibilidad de existencia de varios procesadores en cada estación. Dicho modelo emplea las variables adicionales definidas en el modelo $M2$. A continuación se encuentra

el modelo $M4$:

$$Min \quad W = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \tag{2.29}$$

Sujeto a:

$$\sum_{t=1}^{T} x_{i,t} = d_i \qquad i = 1, ..., |I| \tag{2.30}$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad t = 1, ..., T \tag{2.31}$$

$$\rho_{k,t} = \sum_{i=1}^{|I|} p_{i,k} x_{i,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.32}$$

$$\rho_{k,t} - w_{k,t} \geqslant 0 \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.33}$$

$$\hat{s}_{k,t} \geqslant \hat{s}_{k,t-1} + \rho_{k,t-1} - w_{k,t-1} - c \qquad k = 1, ..., |K|; t = 2, ..., T \tag{2.34}$$

$$\hat{s}_{k,t} \geqslant \hat{s}_{k-1,t} + \rho_{k-1,t} - w_{k-1,t} - c \qquad k = 2, ..., |K|; t = 1, ..., T \tag{2.35}$$

$$\hat{s}_{k,t} + \rho_{k,t} - w_{k,t} \leqslant l_k \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.36}$$

$$\hat{s}_{k,t} \geqslant 0; w_{k,t} \geqslant 0 \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.37}$$

$$x_{i,t} \in \{0, 1\} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.38}$$

$$\hat{s}_{1,1} = 0 \tag{2.39}$$

Fijando $\hat{s}_{1,1} = 0$ como inicio de las operaciones, en el modelo, la función objetivo (2.29) representa la minimización de la sobrecarga total de trabajo; las ecuaciones (2.32) vinculan los tiempos de proceso requeridos por los tipos de producto con los tiempos de proceso requeridos por las unidades de la secuencia; finalmente, las restricciones (2.33) a (2.37) establecen los límites inferior y superior de las sobrecargas y los instantes relativos de inicio de las operaciones; en $M4$ dichos instantes son:

$$\hat{s}_{k,t} = [s_{k,t} - (t + k - 2)c]^{+} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.40}$$

Tabla 2.1: Comparación de las principales diferencias de los modelos $M1$ y $M2$ con $M3$ y $M4$

	$M1$	$M2$	$M3$	$M4$
Función objetivo	$MaxV$	$MinW$	$MaxV$	$MinW$
Instantes de inicio	$s_{k,t}$ absolutos	$\hat{s}_{k,t}$ relativos	$s_{k,t}$ absolutos	$\hat{s}_{k,t}$ relativos
Variables	$v_{k,t}$	$w_{k,t}$	$v_{k,t}$	$w_{k,t}$
Ventana temporal $(t = T)$	$l_k, \forall k$	$c, \forall k$	$l_k, \forall k$	$l_k, \forall k$
Rango para b_k	$b_k \geqslant 1$	$b_k = 1$	$b_k \geqslant 1$	$b_k \geqslant 1$
Vínculos entre estaciones	No	No	Sí	Sí

Las diferencias entre los modelos originales ($M1$ y $M2$) y los propuestos aquí ($M3$ y $M4$) se puede ver con más detalle en (Bautista, Cano and Alfaro, 2012d) y resumido en la tabla 2.1.

2.3.2. Un ejemplo numérico

Con el propósito de ilustrar el problema a través de los modelos anteriormente formulados, presentamos el siguiente ejemplo:
Se dispone de 6 unidades de producto ($T = 6$), de las cuales 3 son de tipo A, 1 es de tipo B y 2 son de tipo C. Las unidades de producto se procesan en 3 estaciones de trabajo ($K = 3$), siendo los tiempos de proceso de cada unidad de tipo de producto (A, B, C) en cada estación (m_1, m_2, m_3) los recogidos en la Tabla 2.2.

Tabla 2.2: Tiempos de proceso ($p_{i,k}$) requeridos por las unidades de producto, según tipo, en cada estación o módulo

	A $(d_A = 3)$	B $(d_B = 1)$	C $(d_C = 2)$
m_1	5	4	3
m_2	5	4	4
m_3	4	3	5
Total	14	11	12

Se considera además: $c = 4$ (tiempo de ciclo) y $l_k = 6$, para todo $k = 1, ..., 3$ (ventanas temporales). La solución óptima dada por el programa lineal mixto asociado al modelo $M1$, considerando $b_k = 1$ para todo $k = 1, ..., 3$, se recoge en la Figura 2.1. Una secuencia de productos que presenta el máximo trabajo total completado es: $A - C - B - A - C - A$. Podemos observar que dicho trabajo es $V = 76$ y la sobrecarga es $W = 1$. Dicha sobrecarga o trabajo perdido se produce en la estación m_2 sobre la segunda unidad secuenciada que es del tipo C.
Por otro lado, la solución dada por el programa lineal mixto asociado al modelo $M2$, se recoge en la Figura 2.2. Una secuencia óptima asociada a dicha solución es: $A-A-A-C-C-B$, presentando un trabajo total completado $V = 72$ y una sobrecarga $W = 5$ que se encuentra repartida entre las estaciones (1 para m_1, 3 para m_2 y 1 para m_3).
Las diferencias en las soluciones óptimas obtenidas a partir de ambos modelos, que en principio son equivalentes, merecen algunos comentarios.
En primer lugar, los instantes de inicio de las operaciones en cada estación son conceptualmente distintos entre los modelos. Mientras que en $M1$, en cada estación y de manera independiente, se tiene en cuenta el instante de finalización de la operación anterior, tomando el instante 0 como referencia de inicio de los trabajos en cada estación; en $M2$, los instantes de inicio de las operaciones se vinculan al inicio de un nuevo ciclo tomando el valor 0 como referencia para representar

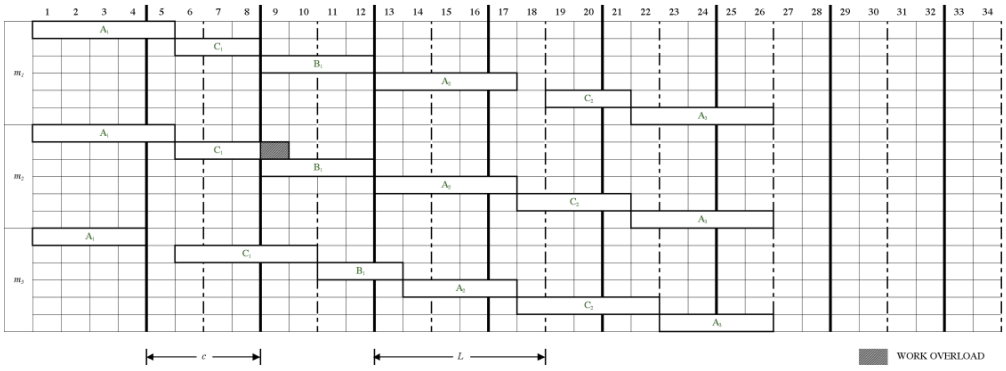

Figura 2.1: Solución óptima dada por $M1$ para el Ejemplo

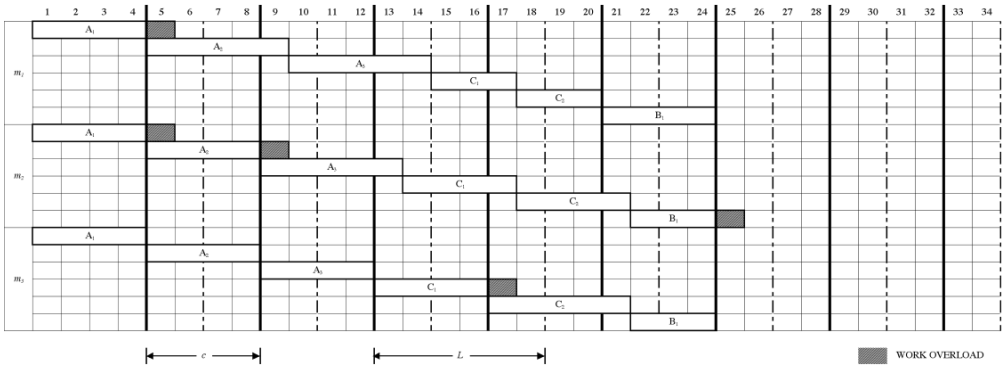

Figura 2.2: Solución óptima dada por $M2$ para el Ejemplo

este momento. Para facilitar la comparación entre ambas soluciones, en las Figuras 2.1 y 2.2 se ha optado por emplear el tipo de instantes de inicio propuestos en el modelo de (Yano and Rachamadugu, 1991).

En segundo lugar, aunque ambos modelos se consideran equivalentes, se puede observar una diferencia entre los valores de la sobrecarga: $W = 1$ para $M1$ y $W = 5$ para $M2$. Esto puede explicarse porque $M1$ considera la posibilidad de utilizar para la última operación de cada estación el tiempo total de ventana (l_k), mientras que $M2$ concede a la última operación de cada estación un tiempo no superior al ciclo c; es fácil ver que, sin esta limitación, $M2$ también ofrece una solución con $W = 1$.

Como se puede observar en las Figuras 2.1 y 2.2, las estaciones no disponen vínculos entre ellas. Si tenemos en cuenta que deseamos tener en cuenta dichos vínculos, las soluciones ofrecidas por los modelos $M1$ y $M2$, teniendo en cuenta que los instantes de inicio de las operaciones de la misma unidad están desplazados un ciclo entre cada para de estaciones vecinas, pueden ofrecer soluciones incoherentes. En las Figuras 2.3 y 2.4 el programa de las operaciones $M1$ y $M2$ se ha desplazado teniendo en cuenta este tiempo de desplazamiento.

En la Figura 2.3 podemos observar varias incoherencias en la programación, por ejemplo en la

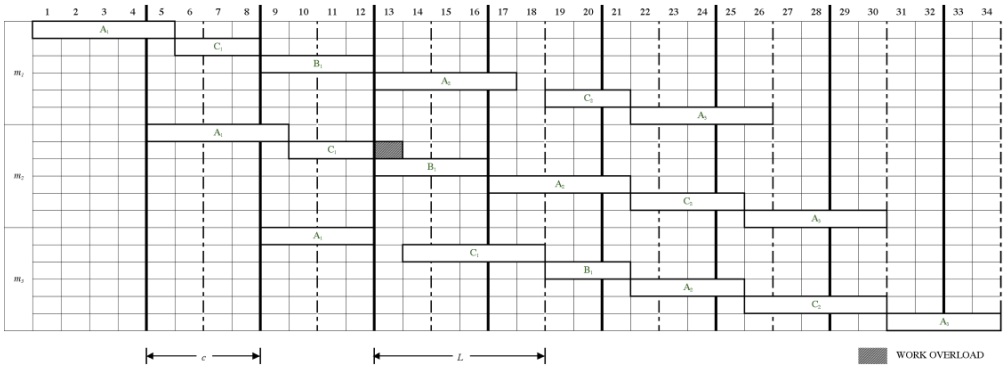

Figura 2.3: Solución desplazada dada por $M1$ para el Ejemplo

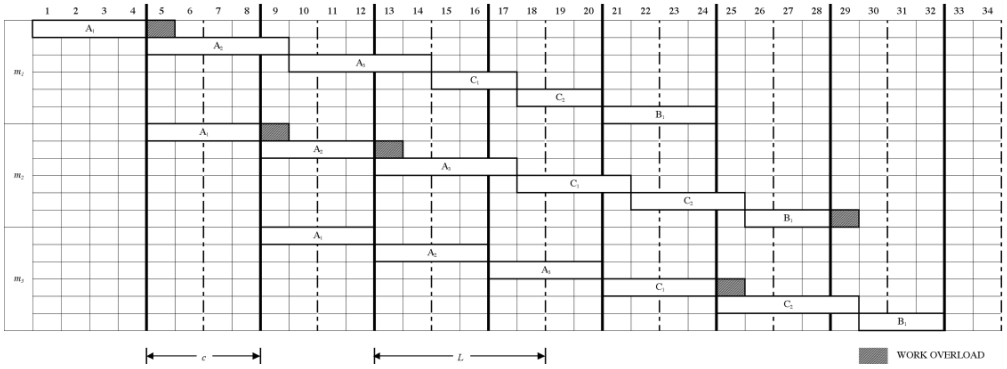

Figura 2.4: Solución desplazada dada por $M2$ para el Ejemplo

transición entre A_1 y A_2 en las estaciones m_1 y m_2. Por otro lado, en la Figura 2.4 también es posible observar incoherencias en la transición de A_2 entre las estaciones m_1 y m_2, y en C_1 entre m_2 y m_3.

Sin embargo, los modelos extendidos para considerar vínculos entre estaciones ($M3$ y $M4$) ofrecen resultados coherentes, como muestran las Figuras 2.5 y 2.6.

A la vista de la solución óptima dada por el programa lineal mixto asociado al modelo $M3$, considerando $b_k = 1$ para todo $k = 1, ..., 3$ (Figura 2.5), la secuencia de productos que presenta el máximo trabajo total completado es: $C - C - A - A - B - A$. Podemos ver que dicho trabajo es $V = 75$ y la sobrecarga es $W = 2$. Dicha sobrecarga o trabajo perdido se concentra sobre unidades del tipo A en las estaciones m_1 y m_2.

Análogamente, en la Figura 2.6, correspondiente a la programación óptima de operaciones dada por $M4$, la secuencia obtenida es: $B - C - A - C - A - A$ (distinta a la de $M3$); la sobrecarga total es $W = 2$ (concentrada en m_2 sobre 2 unidades de tipo A) y el trabajo total completado es $V = 75$.

Las soluciones ofrecidas por $M3$ y $M4$ son óptimas cuando consideramos vínculos entre estaciones consecutivas y son realistas cuando se admiten las siguientes hipótesis de trabajo:

1. Se minimiza la sobrecarga global sin tener en cuenta en qué puntos del desarrollo de los

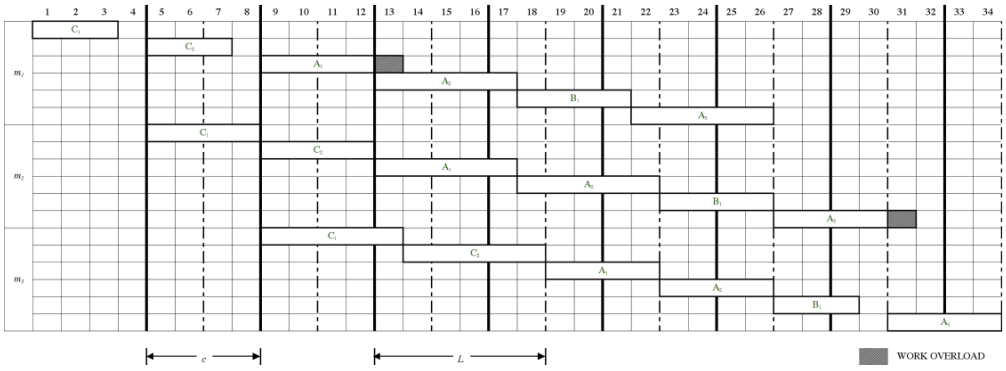

Figura 2.5: Solución óptima dada por $M3$ para el Ejemplo

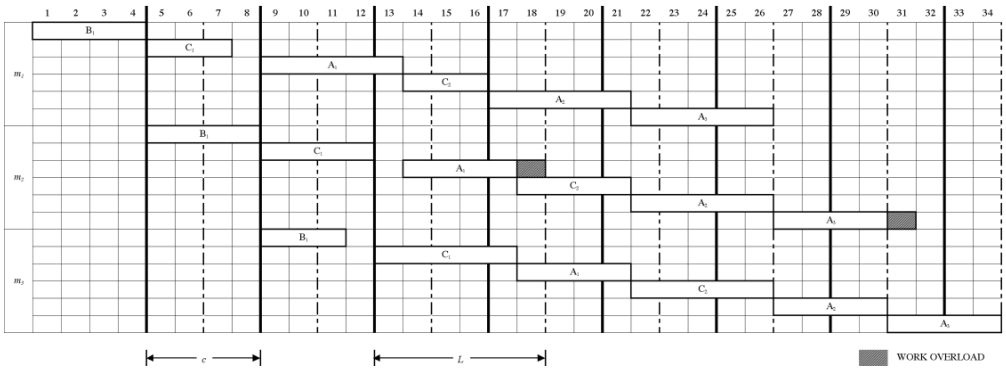

Figura 2.6: Solución óptima dada por $M4$ para el Ejemplo

trabajos y en qué módulos resulta más conveniente o cómoda la presencia de sobrecargas parciales. Se permite la interrupción de la *t-ésima* operación en la estación k en cualquier instante comprendido en el intervalo $[(t+k-1)c, (t+k-2)c+l_k]$.

2. La aparición de sobrecargas parciales implica la intervención sobre las operaciones afectadas, ya sea dejando pendiente una parte del trabajo sobre la operación (siempre que sea posible) o solicitando el concurso de operarios de refuerzo o instrumentación adicional que permita acelerar el tiempo de proceso de dichas operaciones.

3. Para llevar a cabo el trabajo en las estaciones, se dispone de un sistema de control que informa en qué momentos precisos se deben llevar a cabo, o bien, la interrupción del trabajo sobre los productos (aunque no se haya completado) o bien la intervención de los operarios y/o equipos de refuerzo.

Las soluciones recogidas en las Figuras 2.5 y 2.6 cumplirían con las anteriores hipótesis. Por ejemplo, en la Figura 2.5 podemos ver que en las estaciones 1 y 2 se propone intensificar el trabajo sobre las unidades 3 y 6, siendo ambas del tipo A, lo que equivale a incrementar la actividad en un 25 % en ambos casos; por su parte, la solución ofrecida por $M4$ (Figura 2.6) requiere intensificar

la actividad sobre los mismos productos, pero esta vez concentrándose el refuerzo en la estación 2 solamente.

2.3.3. Modelos con vínculos entre estaciones e instantes de interrupción restringidos

Las hipótesis implícitas en los modelos $M3$ y $M4$ son válidas en determinadas líneas de fabricación. No obstante, en algunos entornos reales podemos encontrarnos con serios impedimentos a la hora de intentar aplicar directamente las soluciones ofrecidas por ambos modelos.
En efecto, no siempre podrá llevarse a la práctica la posibilidad de interrumpir cualquier operación en cualquier instante del intervalo comprendido entre el instante final del ciclo en curso y el instante máximo de finalización permitido a través de la ventana temporal. Dicha posibilidad requiere, básicamente, poder informar en los instantes precisos sobre las interrupciones y/o refuerzos para facilitar el trabajo de los operarios y tener programados los autómatas para actuar fuera de la rutina.
Las condiciones anteriores no son imposibles de conseguir, no obstante, aunque requieren una alta automatización y, evidentemente, encarecen de manera notable el sistema de control de las operaciones de producción. Para facilitar la gestión de interrupciones e intensificaciones de trabajo, proponemos una regla muy sencilla: una operación sólo será interrumpida en el instante máximo de finalización marcado por la ventana temporal, si éste se alcanza en la ejecución de la operación. En tales condiciones, para el modelo $M3$, tendremos:

$$e_{k,t} = min \left\{ s_{k,t} + \sum_{i=1}^{|I|} p_{i,k} x_{i,t}, (t+k-2)c + l_k \right\} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.41}$$

donde $e_{k,t}$ es el instante de finalización o interrupción de la *t-ésima* operación en la estación k.
Por otra parte, el trabajo no completado de la *t-ésima* unidad de la secuencia en la estación k se puede formular:

$$w_{k,t} = \sum_{i=1}^{|I|} p_{i,k} x_{i,t} - v_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.42}$$

y el tiempo ocioso será:

$$u_{k,t} = [(t+k-1)c - e_{k,t}]^+ \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.43}$$

y el instante de finalización corregido será:

$$e^c_{k,t} = e_{k,t} + u_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.44}$$

La regla de interrupción propuesta se puede incorporar a $M3$, resultando en el modelo matemático $M5$:

$$Max \quad V = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} v_{k,t} \right) \tag{2.45}$$

Sujeto a:

$$(2.20) - (2.28)\ de\ M3 \tag{2.46}$$

$$v_{k,t} \geqslant \sum_{i=1}^{|I|} p_{i,k} x_{i,t} - M(1 - y_{k,t}) \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.47}$$

$$s_{k,t} + v_{k,t} \geqslant (t + k - 2)c + l_k - M y_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.48}$$

$$y_{k,t} \in \{0, 1\} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.49}$$

donde $y_{k,t}$ es una variable binaria que adopta el valor 1 si el trabajo requerido a la *t-ésima* operación en la estación k se completa íntegramente, y 0 si se produce interrupción; además las restricciones (2.47) y (2.48) permiten fijar el valor de los trabajos completados, $v_{k,t}$, distinguiendo si hay interrupción o no de la operación según la regla propuesta.
En cuanto a aplicar la regla de interrupción propuesta en el modelo $M4$, podemos formular:

$$\hat{e}_{k,t} = max\left\{\hat{s}_{k,t} + \rho_{k,t}, l_k\right\} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.50}$$

donde $\hat{e}_{k,t}$ es el instante de finalización o interrupción relativo al ciclo asociado a la estación k y a la unidad en proceso.
El trabajo completado de la *t-ésima* unidad de la secuencia en la estación k será:

$$v_{k,t} = \rho_{k,t} - w_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.51}$$

el tiempo ocioso se formula:

$$u_{k,t} = [c - \hat{e}_{k,t}]^{+} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.52}$$

y el instante de finalización relativo y corregido es:

$$\hat{e}^{c}_{k,t} = \hat{e}_{k,t} + u_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.53}$$

Añadiendo a $M4$ la regla de interrupción de operaciones propuesta resulta en el siguiente modelo matemático, $M6$:

$$Min \quad W = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \tag{2.54}$$

Sujeto a:

$$(2.30) - (2.39)\ de\ M4 \tag{2.55}$$

$$w_{k,t} \leqslant M(1 - y_{k,t}) \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.56}$$

$$\hat{s}_{k,t} + \rho_{k,t} - w_{k,t} \geqslant l_k - M y_{k,t} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.57}$$

$$y_{k,t} \in \{0, 1\} \qquad k = 1, ..., |K|; t = 1, ..., T \tag{2.58}$$

donde $M = 2c$, y la variable binaria $y_{k,t}$ tiene un significado equivalente al que posee en $M5$: adopta el valor 1 cuando no hay sobrecarga en la *t-ésima* operación en la estación k, y 0 en caso contrario; además las restricciones (2.56) y (2.57) permiten fijar el valor de la sobrecarga en función de la interrupción o no de la operación. En cuanto a las funciones objetivo propuestas para $M5$ y $M6$ (idénticas a las de $M3$ y $M4$, respectivamente), es fácil establecer una relación entre ellas a través de las igualdades (2.42), tal y como sigue:

Teorema 1. *Si $b_k = 1(k = 1, ..., |K|)$, entonces las funciones objetivos de $M3$, $M4$, $M5$ y $M6$ son equivalentes.*

Demostración. Tenemos:

$$Max \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T} v_{k,t}\right) \Leftrightarrow Max \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T}\left(\sum_{i=1}^{|I|} p_{i,k}x_{i,t} - w_{k,t}\right)\right) \Leftrightarrow$$

$$Max \sum_{k=1}^{|K|} b_k \left(\sum_{t=1}^{T}\sum_{i=1}^{|I|} p_{i,k}x_{i,t} - \sum_{t=1}^{T} w_{k,t}\right) \Leftrightarrow Max \left(V_0 - \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T} w_{k,t}\right)\right) \Leftrightarrow$$

$$Min \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T} w_{k,t}\right)$$

donde $V_0 = \sum_{k=1}^{|K|}\left(b_k \sum_{i=1}^{|I|} d_i p_{i,k}\right)$ es el trabajo total requerido.
Finalmente,

$$Max \;\; V = \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T} v_{k,t}\right) \Leftrightarrow Min \;\; W = \sum_{k=1}^{|K|}\left(b_k \sum_{t=1}^{T} w_{k,t}\right) \tag{2.59}$$

Por tanto, para $b_k = 1(k = 1, ..., |K|)$ los objetivos de $M3$, $M4$, $M5$ y $M6$ son equivalentes, y, por tanto, las soluciones que ofrecen $M3$ y $M4$, por una parte, y $M5$ y $M6$, por otra, también deben ser equivalentes. □

Las soluciones óptimas, para el ejemplo anterior, ofrecidas por los modelos $M5$ y $M6$ se recogen en las Figuras 2.7 y 2.8, respectivamente.

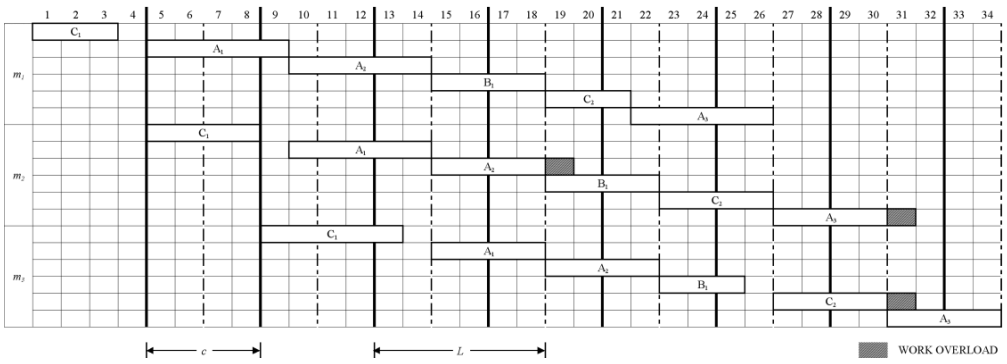

Figura 2.7: Solución óptima dada por $M5$ para el Ejemplo

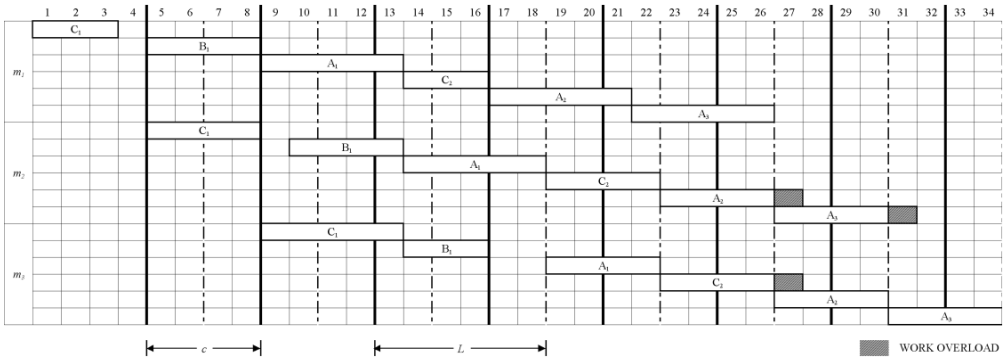

Figura 2.8: Solución óptima dada por $M6$ para el Ejemplo

Los modelos $M5$ y $M6$ ofrecen soluciones óptimas con sobrecarga total $W = 3$, superior en una unidad a la ofrecida por $M3$ y $M4$, pero cumpliendo la regla de interrupción. En las Figuras 2.7 y 2.8 podemos observar que los operarios interrumpirán la operación justo en el instante en que se alcanza el límite impuesto por la ventana temporal. Sobre un trabajo total requerido $V_0 = 77$, se ha completado un trabajo $V = 74$. Por otro lado, de cara a evaluar una secuencia de productos $\pi = (\pi_1, ..., \pi_T)$ sujeta a las restricciones impuestas en $M5$ y $M6$, también sería válida la asignación de valores a las variables $x_{i,t}$:

$$x_{i,t} = \begin{cases} 1 & \text{Si } \pi_t = i, \\ 0 & \text{Si } \pi_t \neq i. \end{cases} \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.60}$$

Sin embargo, para evitar el efecto sobre el tiempo de resolución de las variables binarias $y_{k.t}$, se propone el siguiente procedimiento:

Algoritmo 1 *Proc1*: Determinación de V (trabajo total completado), W (sobrecarga total) y U (tiempo ocioso) para una secuencia $\pi = (\pi_1, ..., \pi_T)$

Entrada: $T, I, K, \pi, b_k\ \forall k, p_{i,k}\ \forall k, \forall t$

Salida: $V, W, U, s_{k,t}, e_{k,t}, v_{k,t}, w_{k,t}, u_{k,t}\ \forall k, \forall t$

1: Inicialización: $t = V = W = U = 0$ (donde U representa el tiempo ocioso total)
2: **mientras** $t \leqslant T$ **hacer**
3: $t \leftarrow t + 1; e_{0,t} = 0$
4: **mientras** $k \leqslant |K|$ **hacer**
5: $k \leftarrow k + 1; e_{k,0} = 0$
6: $s_{k,t} = max\{e_{k,t-1}, e_{k-1,t}, (t + k - 2)c\}$
7: $e_{k,t} = min\{s_{k,t} + p_{\pi,k}, (t + k - 2)c + l_k\}$
8: $v_{k,t} = e_{k,t} - s_{k,t}$
9: $w_{k,t} = p_{\pi,k} - v_{k,t}$
10: $u_{k,t} = [(t + k - 1)c - e_{k,t}]^+$
11: $V \leftarrow V + b_k v_{k,t}$
12: $W \leftarrow W + b_k w_{k,t}$
13: $U \leftarrow U + b_k u_{k,t}$
14: **fin mientras**
15: **fin mientras**

Aunque el problema original ($M1$) es *NP-difícil* (Yano and Rachamadugu, 1991), el algoritmo *Proc1* se puede usar para evaluar V, W y U cuando se dispone de vínculos entre estaciones y instante de interrupción restringidos.

2.3.4. Grafo asociado al problema

Con el fin de aplicar procedimientos basados en programación dinámica, es necesario definir un conjunto de estados para el problema. De una forma similar a (Bautista et al., 1996a), es posible construir un grafo conectado sin bucles ni ciclos directos de $T + 1$ etapas. El conjunto de vértices en el nivel t ($t = 0, ..., T$) lo notaremos como $J(t)$. Sea $J(t, j)$ ($j = 1, ..., |J(t)|$) un vértice del nivel t, éste queda definido por la terna $(\vec{q}(t, j), \vec{e}(t, j), W(t, j))$, donde:

- $\vec{q}(t, j) = (q_1(t, j), ..., q_{|I|}(t, j))$ representa el vector de demanda satisfecha asociada al vértice.
- $\vec{e}(t, j) = (e_1(t, j), ..., e_{|K|}(t, j))$ representa el vector de instantes de finalización, en las estaciones, de las operaciones asociadas al vértice.
- $W(t, j)$ representa el trabajo total no completado (o sobrecarga) sobre las operaciones asociadas al vértice.

Además, el vértice $J(t,j)$ presenta las siguientes propiedades:

$$\sum_{i=1}^{|I|} q_i(t,j) = t \tag{2.61}$$

$$0 \leqslant q_i(t,j) \leqslant d_i \qquad i = 1,...,|I| \tag{2.62}$$

$$e_k^c(t,j) = max\left\{e_k(t,j), (t+k-1)c\right\} \qquad k = 1,...,|K| \tag{2.63}$$

siendo $\vec{e}^c(t,j)$ el vector de instantes de finalización corregidos en consonancia con el tiempo de ciclo.
En definitiva, un vértice $J(t,j)$ lo representaremos de la forma:

$$J(t,j) = \{(t,j), \vec{q}(t,j), W(t,j), \vec{e}(t,j), \vec{e}^c(t,j)\} \tag{2.64}$$

En el nivel 0 del grafo hay un sólo vértice $J(0)$. Inicialmente, podríamos pensar que en el nivel t, $J(t)$ contiene los vértices asociados a todas las subsecuencias que pueden construirse con t productos satisfaciendo las propiedades (2.61) y (2.62), no obstante, es fácil reducir el cardinal que a priori puede presentar $J(t)$ estableciendo las siguientes relaciones de dominancia y equivalencia:

$$J(t,j) \prec J(t,j') \Leftrightarrow \left[\vec{q}(t,j) = \vec{q}(t,j')\right] \wedge \left[\vec{e}^c(t,j) \prec \vec{e}^c(t,j')\right] \wedge \left[W(t,j) \leqslant W(t,j')\right] \tag{2.65}$$

$$J(t,j) \equiv J(t,j') \Leftrightarrow \left[\vec{q}(t,j) = \vec{q}(t,j')\right] \wedge \left[\vec{e}^c(t,j) = \vec{e}^c(t,j')\right] \wedge \left[W(t,j) = W(t,j')\right] \tag{2.66}$$

Con dichas relaciones podremos reducir el espacio de búsqueda de soluciones en el grafo. Por tanto, en el nivel t del grafo, $J(t)$ contendrá los vértices asociados a subsecuencias no dominadas y no equivalentes, y, en el nivel T, $J(T)$ contendrá todos los vértices asociados a secuencias completas no equivalentes y no dominadas.
Entre los vértices $J(t,j)$, del nivel t, y el vértice $J(t+1,j_i)$ del nivel $t+1$, existe un arco o transición, a través del tipo de producto i, en caso que:

$$\vec{q}(t,j) \prec \vec{q}(t+1,j_i) \tag{2.67}$$

En la transición $J(t,j) \xrightarrow{i} J(t+1,j_i)$, se cumple:

$$\vec{i} = \vec{q}(t+1,j_i) - \vec{q}(t,j) \tag{2.68}$$

donde $\vec{i}$ es el vector canónico representante del tipo de producto i.
Para que el vértice $J(t+1,j_i)$ quede completamente definido a través de la transición desde $J(t,j)$ será necesario determinar $J(t+1,j_i) = \{(t+1,j_i), \vec{q}(t+1,j_i), W(t+1,j_i), \vec{e}(t+1,j_i), \vec{e}^c(t+1,j_i)\}$; para ello, se puede emplear el siguiente procedimiento:

Algoritmo 2 *Proc2* $(J(t,j),i)$: Transición $J(t,j) \xrightarrow{i} J(t+1,j_i)$

Entrada: $|K|, t, i, J(t,j); p_{i,k}\ \forall i, \forall k$

Salida: $J(t+1,j_i), \vec{e}(t+1,j_i), W(t+1,j_i), \vec{e}^{c}(t+1,j_i)$

1: Inicialización: $k=0; e_0(t+1,j_i)=0; W(t+1,j_i)=W(t,j)$
2: **mientras** $k \leqslant |K|$ **hacer**
3: $\quad k \leftarrow k+1$
4: $\quad s = max\{e_k(t,j), e_{k-1}(t+1,j_i), (t+k-1)c\}$
5: $\quad e_k(t+1,j_i) = min\{s+p_{i,k}, (t+k-1)c+l_k\}$
6: $\quad e_k^c(t+1,j_i) = max\{e_k(t+1,j_i), (t+k)c\}$
7: $\quad w = s+p_{i,k}-e_k(t+1,j_i)$
8: $\quad W(t+1,j_i) \leftarrow W(t+1,j_i)+b_k w$
9: **fin mientras**
10: $\vec{q}(t+1,j_i) = \vec{q}(t,j)$
11: $q_i(t+1,j_i) = q_i(t,j)+1$

Indirectamente, se puede calcular la aportación a la sobrecarga global generada en la transición de $J(t,j)$ a $J(t+1,j_i)$, a través de la incorporación del producto i a este último vértice, de la forma:

$$a((t,j) \rightarrow (t+1,j_i)) = W(t+1,j_i) - W(t,j) \tag{2.69}$$

En tales condiciones, buscar una secuencia, que optimice los objetivos equivalentes propuestos en (2.59), es equivalente a encontrar un camino óptimo (máximo para V y mínimo para W) desde el vértice $J(0)$ hasta el conjunto de vértices $J(T)$ de la etapa T.

Por tanto, cualquier algoritmo de caminos extremos en grafos es válido para encontrar soluciones al problema propuesto. Sin embargo, los problemas realistas industriales, donde $|I|$ y T son de gran dimensión, darán lugar a grafos con un número de vértices muy grande, por lo que será conveniente recurrir a procedimientos que no requieran explícitamente, para el cálculo, la presencia de todos los vértices.

Obviamente, para hallar soluciones al problema, también se pueden emplear heurísticas:

1. Greedy constructivas que compongan una secuencia, paso a paso, mediante las aportaciones (2.69), o similares, obtenidas a través de *Proc2*.

2. Heurísticas basadas en *local search* con la asistencia de *Proc1* para evaluar vecindarios.

2.3.5. Acotación de los valores de las secuencias

Para el uso de procedimientos heurísticos basados en programación dinámica acotada, es necesario definir cotas para el problema. En este caso, en primer lugar estableceremos cotas globales para la función W definida en (2.59) y, posteriormente, estableceremos cotas parciales asociadas al camino por construir (complemento) cuando un segmento o subsecuencia de t elementos ya ha sido construido.

Cotas globales para W

Si tenemos en cuenta las estaciones de trabajo de manera independiente, podemos escribir:

$$LB1(k) = b_k \left[\sum_{i=1}^{|I|} d_i p_{i,k} - (T-1)c - l_k \right]^+ \qquad k = 1, ..., |K| \tag{2.70}$$

la cual es una cota del trabajo que no se puede concluir en la estación k.
Por tanto, considerando todas las estaciones, tendremos:

$$LB1 = \sum_{k=1}^{|K|} LB1(k) \tag{2.71}$$

Análogamente, también podemos considerar de forma independiente el trabajo requerido para cada tipo de producto, obteniendo:

$$LB2(i) = \left[\sum_{k=1}^{|K|} b_k p_{i,k} - c \sum_{k=1}^{|K|} b_k - b_{|K|}(l_{|K|} - c) \right]^+ \qquad i = 1, ..., |I| \tag{2.72}$$

y considerando las demandas de los productos, tendremos:

$$LB2 = \sum_{i=1}^{|I|} d_i LB2(i) \tag{2.73}$$

Otra alternativa para obtener cotas a partir de los productos consiste en emplear el procedimiento *Proc2*:

$$LB3(i) = W(1,i) \qquad i = 1, ..., |I| \tag{2.74}$$

$$W(1,i) \leftarrow Proc2(J(0) \xrightarrow{i} J(1,i)) \qquad i = 1, ..., |I| \tag{2.75}$$

y la cota global de W será:

$$LB3 = \sum_{i=1}^{|I|} d_i LB3(i) \tag{2.76}$$

Cota del complemento de una subsecuencia dada

Supongamos que hemos construido un camino desde $J(0)$ hasta el vértice $J(t,j)$ y disponemos, por tanto, de la información $W(t,j), \vec{q}(t,j), \vec{e}(t,j)$ y $\vec{e}^c(t,j)$.
Para completar una secuencia hasta la etapa T, necesitamos enlazar con $J(t,j)$, $T-t$ vértices $(J(t+1,j_1), J(t+2,j_2), ..., J(T,j_{T-t}))$. Los valores asociados a estas transiciones nos darán un complemento R, que sumado a $W(t,j)$, nos dará un valor para W. Haciendo $j = j_0$, podemos escribir:

$$R(t,j) = \sum_{\tau=0}^{T-t-1} a((t+\tau, j_\tau) \rightarrow (t+\tau+1, j_{\tau+1})) \tag{2.77}$$

$$W = W(t,j) + R(t,j) \tag{2.78}$$

En tales condiciones, podemos acotar $R(t,j)$ adaptando la cota global $LB1$.
En efecto, dada la estación k, el tiempo disponible para realizar las operaciones pendientes es:

$$TD_k(t,j) = (T+k-2)c + l_k - e_k^c(t,j) \qquad k = 1,...,|K| \tag{2.79}$$

y el tiempo requerido para realizar dichas operaciones es:

$$TP_k(t,j) = \sum_{i=1}^{|I|}(d_i - q_i(t,j))p_{i,k} \qquad k = 1,...,|K| \tag{2.80}$$

En consecuencia:

$$LB1(t,j) = \sum_{k=1}^{|K|}[TP_k(t,j) - TD_k(t,j)]^+ \tag{2.81}$$

es una cota inferior del complemento $R(t,j)$.
Si atendemos a los tipos de producto, tendremos:

$$LB2(t,j) = \sum_{i=1}^{|I|}(d_i - q_i(t,j))LB2(i)) \tag{2.82}$$

y:

$$LB3(t,j) = \sum_{i=1}^{|I|}(d_i - q_i(t,j))LB3(i)) \tag{2.83}$$

que también son cotas del complemento $R(t,j)$.

2.3.6. El uso de la programación dinámica acotada

El procedimiento que proponemos (ver (Bautista et al., 1996a) y (Bautista and Pereira, 2009)), programación dinámica acotada (*BDP*), consiste en generar, nivel a nivel, desde el nivel 0 hasta el nivel T, una parte del grafo descrito en la sección 2.3.4.
Los vértices generados forman, potencialmente, parte de un camino óptimo (de 0 a T), basándonos en la construcción de un segmento óptimo de t etapas, desde $J(0)$ hasta $J(t,j)$, y en la evaluación de una cota del complemento $R(t,j)$ para alcanzar la etapa T, por ejemplo $LB1(t,j)$.
El procedimiento sólo mantiene en memoria la información de dos etapas consecutivas, t y $t+1$ ($t = 0,...,T-1$), empleando para ello las listas $\Lambda(t)$ y $\Lambda(t+1)$, respectivamente:

- La lista $\Lambda(t)$ contiene la información sobre los vértices consolidados en la etapa t que, potencialmente, pueden formar parte de un camino óptimo o de buena calidad.
- La lista $\Lambda(t+1)$ contiene los vértices que, tentativamente, se generan de uno en uno a partir de cada vértice de la lista $\Lambda(t)$, mediante las transiciones posibles entre las etapas t y $t+1$.

Un registro $\lambda(J(t,j))$ de la lista $\Lambda(t)$, $\lambda(J(t,j)) \in \Lambda(t)$, está formado por tres elementos:

$$\lambda(J(t,j)) = \left\{J(t,j), LB1(t,j), \Gamma^{-}(J(t,j))\right\} \tag{2.84}$$

donde $\Gamma^{-}(J(t,j))$ es el vértice de la etapa $t-1$ progenitor de $J(t,j)$.
Aunque el empleo de $\Lambda(t)$ y $\Lambda(t+1)$ reduce notablemente las necesidades de memoria, el número de vértices que se pueden generar para una etapa puede ser muy elevado. Por ello, imponemos una limitación sobre el número de vértices $H(t)$ guardados en la etapa t; dicha limitación la notaremos por H y la denominamos ancho de ventana; esto es: $H(t) \leqslant H(t = 1, ..., T)$. Por otra parte, fijamos el número máximo de transiciones a partir de un vértice al valor $|I|$, que es, además, el valor máximo posible.
Evidentemente, algunos vértices generados, tentativamente, en la etapa t no quedarán registrados en la lista $\Lambda(t+1)$. En efecto:

1. Eliminaremos un vértice generado $J(t+1, j_i)$ cuando el valor de su cota inferior, $LBZ = W(t+1, j_i) + LB1(t+1, j_i)$, sea mayor o igual que el valor de una solución Z_0 conocida, ya que a través de $J(t+1, j_i)$ no será posible obtener una solución con mejor valor que Z_0.

2. Rechazaremos un vértice generado $J(t+1, j_i)$ cuando exista un registro $\lambda(J(t+1, h)) \in \Lambda(t+1)$ cuyo vértice domine o sea equivalente a $J(t+1, j_i)$: $J(t+1, h)(\prec \vee \equiv)J(t+1, j_i)$.

3. Desestimamos la entrada de un vértice generado, $J(t+1, j_i)$, a la lista $\Lambda(t+1)$ cuando ésta esté llena ($H(t+1) = H$) y $J(t+1, j_i)$ tenga una cota inferior, $LBZ = W(t+1, j_i) + LB1(t+1, j_i)$, mayor o igual que la mayor de las cotas inferiores de los vértices ya registrados en $\Lambda(t+1)$, aunque a través de $J(t+1, j_i)$ pueda pasar un camino óptimo.

4. El vértice generado, $J(t+1, j_i)$, desplazará a un vértice, $J(t+1, h)$, registrado en la lista $\Lambda(t+1)$, cuando $J(t+1, j_i)$ domine a $J(t+1, h)$; o bien, cuando $J(t+1, j_i)$ tenga menor cota inferior que $J(t+1, h)$ y $H(t+1) = H$, aunque a través del vértice desplazado pueda pasar el camino óptimo.

Para obtener una solución inicial con valor Z_0 (cota superior del valor de la solución óptima), bastará con emplear un procedimiento greedy constructivo asistido por *Proc2*, o emplear local search asistido por *Proc1*, o emplear *BDP* con ancho de ventana pequeño, por ejemplo $H = 1$.
En tales condiciones, podemos escribir el siguiente algoritmo:

Algoritmo 3 *BDP para el problema MMSP-W con instantes de interrupción restringidos*

Entrada: $T, |I|, |K|, d_i(i = 1, ..., |I|), p_{i,k}(i = 1, ..., |I|; k = 1, ..., |K|), l_k, b_k(\forall k), c, Z_0, H$

Salida: $\Lambda(t)$

Inicialización: $t = 0; \Lambda(t) = \{J(0), LB1, -\}; LBZ_{min} = \infty$
mientras $t < T$ **hacer**
  $\Lambda(t+1) = \{\emptyset\}; j = 1$
  mientras $j \leqslant H(t)$ **hacer**
    $i = 1$
    mientras $i \leqslant |I|$ **hacer**
      si $q_i(t, j) < d_i$ **entonces**
        Generar vértice
        $J(t+1, j_i) \leftarrow Proc2(J(t, j), i)$
        Determinar $LB1(t+1, j_i)$ (ver (2.81))
        $LBZ = W(t+1, j_i) + LB1(t+1, j_i)$
        si $LBZ \geqslant Z_0$ **entonces**
          Eliminar $J(t+1, j_i)$
        si no, si $\exists\lambda(J(t+1, h)) \in \Lambda(t+1)/J(t+1, h)(\prec \vee \equiv)J(t+1, j_i)$ **entonces**
          Rechazar $J(t+1, j_i)$
        si no
          Sea $L(t+1) = \{\lambda(t+1, h) \in \Lambda(t+1) : J(t+1, j_i) \prec J(t+1, h)\}$
          si $L(t+1) \neq \{\emptyset\}$ **entonces**
            $\Lambda(t+1) \leftarrow \Lambda(t+1) - L(t+1); H(t+1) \leftarrow H(t+1) - |L(t+1)|$
          fin si
          si $H(t+1) < H$ **entonces**
            $\Lambda(t+1) \leftarrow \Lambda(t+1) + \{J(t+1, j_i), LB1(t+1, j_i), J(t, j)\}$
            $H(t+1) \leftarrow H(t+1) + 1$
          si no
            $\lambda(t+1, h_{max}) = \arg \underset{\lambda(t+1,h)\in\Lambda(t+1)}{\max}(W(t+1, h) + LB1(t+1, h))$
            $LBZ_{max} = W(t+1, h_{max}) + LB1(t+1, h_{max})$
            si $LBZ_{max} \leqslant LBZ$ **entonces**
              Desestimar $J(t+1, j_i)$
            si no
              Desplazar $J(t+1, h_{max})$ con $J(t+1, j_i)$: $\Lambda(t+1) \leftarrow \Lambda(t+1) - \lambda(t+1, h_{max})$
              $\Lambda(t+1) \leftarrow \Lambda(t+1) + \{J(t+1, j_i), LB1(t+1, j_i), J(t, j)\}$
              si $LBZ_{min} > LBZ_{max}$ **entonces**
                $LBZ_{min} = LBZ_{max}$
              fin si
            fin si
          fin si
        fin si
      fin si
      $i \leftarrow i + 1$
    fin mientras
    $j \leftarrow j + 1$
  fin mientras
  Salvar $\Lambda(t); \Lambda(t) \leftarrow \Lambda(t+1); t \leftarrow t + 1$
fin mientras
Salvar $\Lambda(T)$

Cuando el procedimiento finaliza podemos encontrarnos, inicialmente, ante dos situaciones:

- La lista $\Lambda(T)$ está vacía: significa que no hemos podido hallar una solución con valor inferior a Z_0.
- La lista $\Lambda(T)$ no está vacía, significa que los registros contenidos en $\Lambda(T)$, $\lambda(T,h) \in \Lambda(T)$, están asociados a vértices, $J(T,h)$, cuya sobrecarga $W(T,h) < Z_0$; entonces podemos reconstruir, regresivamente, una secuencia con mejor valor que Z_0 a partir de cualquiera de éstos vértices, empleando las listas $\Lambda(t)$ y los predecesores de los vértices.

Además, podemos garantizar que podemos construir una secuencia óptima a partir de los registros $\lambda(T,h) \in \Lambda(T) \neq \{\emptyset\}$ cuando se produce alguno de los casos siguientes:

$$\text{Caso 1: } \max_{0 \leqslant t \leqslant T} \{H(t)\} < H \tag{2.85}$$

$$\text{Caso 2: } \left(\max_{0 \leqslant t \leqslant T} \{H(t)\} = H \right) \wedge (W(T,h) \leqslant LBZ_{min}) \tag{2.86}$$

En cualquier otro caso, el procedimiento es un heurístico.
Aunque se podría desarrollar cualquier tipo de heurísticas para solucionar el problema representado por los modelos $M5$ y $M6$, se ha escogido *BDP* por varias razones:

- *BDP* tiene el potencial de reducir el espacio de búsqueda de soluciones a través del uso de (a) relaciones de dominancia y equivalencia entre vértices y (b) cotas inferiores para los valores de la secuencia y el complemento de un segmento dado.
- Hay disponible una buena guía (LBZ) para encontrar soluciones.
- El esfuerzo dedicado a la búsqueda de una solución (definido por el parámetro H) puede ser convertido fácilmente a tiempo de CPU. Dicha relación permite obtener una nueva solución (ajustando el tiempo de CPU a través de H) bajo circunstancias inesperadas dentro del tiempo disponible (por ejemplo, falta de componentes para un motor o paradas forzadas de la línea de motores).

La Figura 2.9 ilustra la aplicación del algortimo propuesto basado en *BDP* al ejemplo anterior, usando $Z_0 = 4$ y $H = 10$. En el grafo asociado a la Figura 2.9 se puede observar:

1. En la etapa $t = 1$, el vértice asociado con la subsecuencia (B) tiene dos extensiones, (B,A) y (B,C), dominadas por las subsecuencias de la etapa $t = 2$, (A,B) y (C,B), respectivamente. Las extensiones de los vértices: (A,B) de la etapa $t = 2$, (A,A,B), (A,C,B), (C,A,B) y (C,B,C) de la etapa $t = 3$, (A,C,A,B), (C,A,A,B), (C,A,C,B) y (C,C,A,B) de la etapa $t = 4$, y (C,A,A,A,B), (C,A,A,C,B) y (C,C,A,A,B) de la etapa $t = 5$ también están dominadas.

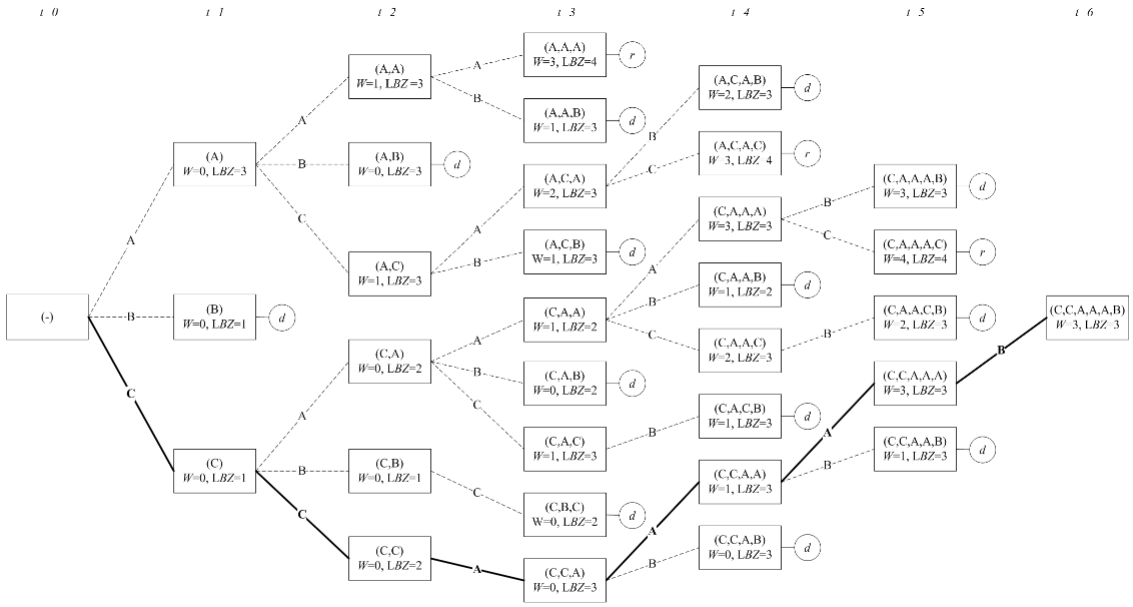

Figura 2.9: Grafo para el ejemplo. En cada vértice se puede observar los siguientes datos: la subsecuencia de productos, el valor de W asociado con la subsecuencia y la cota inferior de la sobrecarga total (LBZ). Las abreviaciones '*d*' y '*r*' simbolizan '*dominado*' y '*eliminado*', respectivamente.

2. En la etapa $t = 3$, el vértice (A, A, A) es eliminado a causa que $LBZ = Z_0 = 4$. Similarmente, los vértices (A, C, A, C) de la etapa $t = 4$ y (C, A, A, A, C) de la etapa $t = 5$ también son eliminados por la misma razón.

3. En la etapa $t = 6$, el vértice (C, C, A, A, A, B) representa la secuencia óptima con valor $W = 3$; siguiendo la traza del camino óptimo (marcado en el grafo) desde $t = 0$ hasta $t = 6$, es posible observar que las sobrecargas parciales están concentradas en los vértices (C, C, A, A) y (C, C, A, A, A).

2.3.7. Experiencia computacional

Para este problema se han llevado a cabo dos experiencias computacionales. La primera de ellas, se realiza sobre ejemplares de referencia que se pueden encontrar en la literatura. La segunda, en cambio, esta basada en ejemplares pertenecientes a la línea de montaje de motores de Nissan ubicada en Barcelona.

Experiencia computacional con instancias de referencia

Se ha realizado una primera prueba de explotación de $M5$ y *BDP* con 225 ejemplares de dimensiones reducidas (ver (Bautista and Cano, 2008)). Estos ejemplares se construyen a partir de 45 planes de demanda (Tabla 2.3) y 5 estructuras de tiempos de proceso requeridos por los productos en las

máquinas (Tabla 2.4).
Dado que no existen soluciones óptimas para dichas instancias considerando vínculos entre estaciones e instantes de interrupción condicionados, para la obtención de las soluciones óptimas hemos empleado dos vías de resolución:

1. Solver CPLEX v11.0 (licencia 1 sólo procesador) ejecutándose en un ordenador MacPro con CPU Intel Xeon 3 GHz y 2 Gb de RAM en sistema operativo Windows XP, sin límite de tiempo de CPU. Los resultados sobre W y tiempos de CPU (resolviendo el modelo $M5$) se recogen detalladamente en la Tabla 2.5. Estos valores son los que se usan como valores de referencia para contrastar la calidad del procedimiento *BDP*.

2. *BDP* programada en gcc v. 4.01 ejecutándose en un ordenador Macintosh iMac con un procesador Intel Core 2 Duo, 2.33 Ghz. y 3 Gb de memoria RAM usando MAC OS X 10.4.11 como sistema operativo. Ni la implementación ni el compilador hacen uso de *threads* ni usa otra forma de código paralelo, y, por tanto, el ordenador usado puede ser visto como un único procesador a 2.33 GHz. Los 225 ejemplares se han resuelto empleando 6 anchos de ventana correlativos, H_1 a H_6, con valores 1, 4, 16, 64, 256 y 512, respectivamente. Hemos fijado al valor 4 el número máximo de transiciones desde cada vértice, haciéndolo coincidir con $|I|$.

En cuanto al valor de la solución inicial, Z_0, hemos empleado para cada ancho de ventana H_α ($\alpha = 1, ..., 6$), el valor de la solución obtenida con el ancho anterior $H_{\alpha-1}$, excepto para el caso con ancho $H_1 = 1$ en el que se fijó Z_0 a ∞.
Se ha recogido en la Tabla 2.6 el número de óptimos hallados por la *BDP*, para cada H_α, sobre los 225 ejemplares y los tiempos medio y máximo empleados. En la columna *#opt BDP* de la Tabla 2.6 aparece el número de óptimos demostrados con el procedimiento *BDP*, para cada ancho de ventana H; mientras que en la columna *#opt* aparece el número de óptimos alcanzados cuando se conocen los valores de éstos (p.e. a través de CPLEX).
En la Tabla 2.6 podemos observar que se alcanzan todos los óptimos con $H = 256$, empleándose en promedio y como máximo, por instancia, unos tiempos de $0{,}079s$ y $0{,}274s$, respectivamente. No obstante, fue necesario emplear una $H = 512$ para garantizar todos los óptimos; en este caso, dichos tiempos fueron de $0{,}0887s$ y $0{,}4194s$.
Nótese el contraste de los tiempos de CPU requeridos por *BDP* en contraposición con CPLEX (ver Tabla 2.5). En efecto, CPLEX empleó un tiempo de CPU promedio, por ejemplar, de $4347{,}76s$ en un rango de $0{,}2s$ a $65359{,}875s$.

Tabla 2.3: Datos experimento: conjunto de 45 programas de producción agrupados en 5 bloques

i	*Block 1*				*Block 2*						*Block 3*							*Block 4*			
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}
d_1	13	1	1	1	7	7	7	1	1	1	5	5	5	3	3	3	4	5	5	5	1
d_2	1	13	1	1	7	1	1	7	7	1	5	3	3	5	3	3	4	5	5	1	5
d_3	1	1	13	1	1	7	1	7	1	7	3	5	3	5	5	4	4	5	1	5	5
d_4	1	1	1	13	1	1	7	1	7	7	3	3	5	3	5	4	4	1	5	5	5

i	*Block 5*																							
	P_{22}	P_{23}	P_{24}	P_{25}	P_{26}	P_{27}	P_{28}	P_{29}	P_{30}	P_{31}	P_{32}	P_{33}	P_{34}	P_{35}	P_{36}	P_{37}	P_{38}	P_{39}	P_{40}	P_{41}	P_{42}	P_{43}	P_{44}	P_{45}
d_1	1	1	1	1	1	1	3	3	3	3	3	3	5	5	5	5	5	5	7	7	7	7	7	7
d_2	3	3	5	5	7	7	1	1	5	5	7	7	1	1	3	3	7	7	1	1	3	3	5	5
d_3	5	7	3	7	3	5	5	7	1	7	1	5	3	7	1	7	1	3	3	5	1	5	1	3
d_4	7	5	7	3	5	3	7	5	7	1	5	1	7	3	7	1	3	1	5	3	5	1	3	1

Tabla 2.4: Datos experimento: 5 estructuras de tiempos de proceso por producto y estación

	Tiempos de proceso para los productos en las estaciones																			
i	*Estructura 1*				*Estructura 2*				*Estructura 3*				*Estructura 4*				*Estructura 5*			
	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4	m_1	m_2	m_3	m_4
p_1	92	103	101	95	91	120	90	100	111	120	85	82	113	119	115	116	115	99	104	96
p_2	97	98	105	104	80	105	113	107	114	113	100	94	114	113	112	118	104	119	100	102
p_3	103	104	99	96	107	88	117	86	83	85	115	119	82	85	84	87	89	98	114	87
p_4	108	95	95	105	114	87	100	114	98	87	110	115	95	87	94	81	95	87	85	118
l_k	108	105	106	106	115	120	120	115	115	120	115	120	115	120	115	120	115	120	115	118
b_k	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tabla 2.5: Soluciones óptimas para las 225 instancias (45 programas x 5 estructuras) usando el modelo $M5$. Las instancias se nombran mediante los número de programa y estructura, separados por '/'. V_0, V, W y CPU simbolizan el trabajo solicitado, el trabajo completado, la sobrecarga y el tiempo de CPU empleado, respectivamente.

Instancia	V_0	V	W	CPU	*Instancia*	V_0	V	W	CPU	*Instancia*	V_0	V	W	CPU
1/1	6292	6243	49	11,516	16/1	5605	5567	38	1300.33	31/1	6410	6352	58	3723.75
1/2	6431	6185	246	5.44	16/2	5670	5535	135	172.44	31/2	6429	6229	200	1959.02
1/3	6407	5996	411	10.8	16/3	5705	5474	231	407.13	31/3	6523	6252	271	1210.91
1/4	7171	6294	877	15.52	16/4	5540	5300	240	333.16	31/4	6397	6083	314	453.67
1/5	6580	6337	243	5.58	16/5	5609	5515	94	250.55	31/5	6468	6288	180	2040.94
2/1	6448	6339	109	9.55	17/1	6400	6360	40	24989.86	32/1	6418	6358	60	362.23
2/2	6479	6113	366	11.17	17/2	6476	6322	154	1561.61	32/2	6511	6287	224	1520.17
2/3	6683	6254	429	15.86	17/3	6524	6258	266	3940.27	32/3	6593	6264	329	3315.09
2/4	7099	6293	806	14.33	17/4	6460	6121	339	24376.44	32/4	6711	6202	509	1219.49
2/5	6712	6350	362	30.8	17/5	6448	6320	128	3118.99	32/5	6530	6307	223	21462.22
3/1	6424	6314	110	9.45	18/1	6388	6331	57	8890.77	33/1	6414	6351	63	26434.95
3/2	6395	6108	287	10.39	18/2	6435	6247	188	10101.59	33/2	6443	6215	228	2431
3/3	6455	6010	445	3.23	18/3	6515	6224	291	1094.3	33/3	6561	6248	313	10045.58
3/4	5671	5591	80	0.23	18/4	6647	6170	477	3528.63	33/4	6635	6169	466	979.42
3/5	6268	6102	166	1.31	18/5	6520	6317	203	3112.27	33/5	6542	6319	223	5813.2
4/1	6436	6285	151	12.72	19/1	6392	6354	38	320.69	34/1	6386	6347	39	441.64
4/2	6599	6259	340	33.48	19/2	6503	6316	187	372.23	34/2	6509	6330	179	3476.92
4/3	6551	6214	337	10	19/3	6547	6236	311	3605.67	34/3	6487	6230	257	540.78
4/4	5899	5819	80	0.27	19/4	6723	6202	521	1176.39	34/4	6285	6033	252	86.53
4/5	6232	6020	212	0.27	19/5	6508	6302	206	27071.11	34/5	6354	6250	104	315.27
5/1	6370	6303	67	5688.09	20/1	6384	6348	36	192.69	35/1	6382	6339	43	215.67
5/2	6455	6159	296	457.59	20/2	6475	6326	149	2798.14	35/2	6441	6299	142	1237.99
5/3	6545	6128	417	943.59	20/3	6471	6214	257	226.83	35/3	6455	6180	275	261.83
5/4	7135	6294	841	9.48	20/4	6247	5995	252	58.86	35/4	6209	5957	252	174.52
5/5	6646	6350	296	647.83	20/5	6360	6284	76	69.47	35/5	6366	6272	94	62.06
6/1	6358	6312	46	5.53	21/1	6436	6351	85	65359.88	36/1	6390	6355	35	39.24
6/2	6413	6299	114	90.27	21/2	6491	6233	258	3861.78	36/2	6523	6341	182	1515.97
6/3	6431	6204	227	47.67	21/3	6563	6262	301	226.52	36/3	6525	6252	273	10945.8
6/4	6421	6083	338	229.25	21/4	6223	5995	228	70.61	36/4	6523	6146	377	3094.97
6/5	6424	6280	144	19.84	21/5	6404	6271	133	3678.36	36/5	6428	6257	171	6617.72
7/1	6364	6350	14	4.94	22/1	6434	6331	103	15804.28	37/1	6384	6332	52	15772.78
7/2	6515	6347	168	90.08	22/2	6511	6228	283	12042.48	37/2	6421	6267	154	1702.38
7/3	6479	6224	255	4852.97	22/3	6541	6204	337	89.8	37/3	6477	6228	249	821.23
7/4	6535	6146	389	3076.41	22/4	6023	5869	154	3.63	37/4	6409	6083	326	853.67
7/5	6406	6240	166	1858.02	22/5	6324	6229	95	284.56	37/5	6446	6290	156	1490
8/1	6436	6369	67	3659.84	23/1	6432	6340	92	2893.02	38/1	6394	6339	55	4662.86
8/2	6437	6168	269	97.84	23/2	6477	6206	271	14503.19	38/2	6483	6241	242	1654.64
8/3	6569	6276	293	83.75	23/3	6525	6170	355	179.69	38/3	6569	6204	365	1541.61
8/4	6385	6083	302	85.67	23/4	5985	5831	154	4.2	38/4	6923	6248	675	1613.24
8/5	6490	6272	218	441.52	23/5	6330	6244	86	427.97	38/5	6588	6329	259	27950.77
9/1	6442	6347	95	2341.7	24/1	6438	6341	97	58324.36	39/1	6392	6327	65	7181.34
9/2	6539	6267	272	312.27	24/2	6525	6254	271	4241.22	39/2	6449	6203	246	6446.81
9/3	6617	6303	314	2031.69	24/3	6579	6277	302	821.2	39/3	6553	6202	351	1802.11
9/4	6499	6145	354	1845.88	24/4	6261	6033	228	122.36	39/4	6885	6234	651	1869.03
9/5	6472	6271	201	247.2	24/5	6398	6251	147	1349.44	39/5	6594	6337	257	2673.03
10/1	6430	6320	110	145.92	25/1	6434	6360	74	6374.41	40/1	6362	6340	22	28.42
10/2	6497	6191	306	1376.95	25/2	6457	6209	248	1372.11	40/2	6481	6349	132	417.55
10/3	6503	6112	391	5.08	25/3	6547	6228	319	114.44	40/3	6463	6228	235	1662.77
10/4	5785	5705	80	0.2	25/4	6185	5957	228	13.05	40/4	6497	6132	365	2625.64
10/5	6250	6203	47	4.08	25/5	6410	6276	134	288.14	40/5	6412	6288	124	658.33
11/1	6390	6345	45	10436.38	26/1	6440	6359	81	9575.78	41/1	6360	6326	34	131.31
11/2	6469	6292	177	2337.83	26/2	6505	6237	268	1164.69	41/2	6447	6331	116	360.63
11/3	6531	6234	297	4963.89	26/3	6601	6297	304	2506.94	41/3	6447	6216	231	242.25
11/4	6685	6188	497	2764.61	26/4	6461	6129	332	1044.7	41/4	6459	6110	349	238.91
11/5	6514	6340	174	3451.33	26/5	6478	6291	187	12453.34	41/5	6418	6296	122	309.66
12/1	6386	6346	40	11612.37	27/1	6438	6369	69	15144.13	42/1	6366	6339	27	626.19
12/2	6455	6315	140	2341.53	27/2	6471	6207	264	995.66	42/2	6495	6320	175	1056.34
12/3	6493	6240	253	1888.22	27/3	6585	6287	298	235.38	42/3	6501	6192	309	20323.33
12/4	6447	6110	337	11290.49	27/4	6423	6109	314	385.69	42/4	6735	6202	533	2384.3
12/5	6440	6314	126	2370	27/5	6484	6305	179	1499.64	42/5	6486	6295	191	6794.7
13/1	6388	6357	31	3542.86	28/1	6408	6334	74	3455.59	43/1	6362	6311	51	6547.89
13/2	6489	6339	150	1483.39	28/2	6503	6281	222	28641.3	43/2	6427	6271	156	1400.38
13/3	6509	6252	257	10937.13	28/3	6495	6180	315	70.55	43/3	6469	6186	283	1266.83
13/4	6485	6132	353	4886	28/4	6035	5869	166	8.52	43/4	6659	6170	489	11708.77
13/5	6434	6303	131	8910.48	28/5	6302	6232	70	40.89	43/5	6498	6314	184	918.8
14/1	6412	6363	49	12299.7	29/1	6406	6342	64	417.75	44/1	6368	6321	47	50178.45
14/2	6463	6276	187	7299.72	29/2	6469	6255	214	49981.44	44/2	6475	6253	222	1802.03
14/3	6539	6264	275	16845.02	29/3	6479	6146	333	219.98	44/3	6523	6160	363	1502.11
14/4	6435	6110	325	1595.95	29/4	5997	5831	166	8.56	44/4	6935	6248	687	537.95
14/5	6462	6318	144	6620.83	29/5	6308	6247	61	23.49	44/5	6566	6327	239	11194.28
15/1	6410	6354	56	20883.03	30/1	6416	6351	65	3957.08	45/1	6366	6307	59	6168.03
15/2	6483	6302	181	2548.69	30/2	6531	6320	211	3098.97	45/2	6441	6215	226	7859.7
15/3	6517	6238	279	333.67	30/3	6571	6280	291	5345.5	45/3	6507	6164	343	537.42
15/4	6235	5995	240	161.05	30/4	6511	6146	365	1680.27	45/4	6897	6234	663	676.22
15/5	6382	6283	99	1989.11	30/5	6450	6269	181	4646.91	45/5	6572	6335	237	1221.91

Tabla 2.6: Número de óptimos alcanzados (*#opt*) y demostrados con *BDP* (*#opt BDP*), para cada ancho de ventana H, sobre los 225 ejemplares. 'promedio CPU average' y 'max CPU' simbolizan los tiempos de CPU, en segundos, medio y máximo empleados.

H	*#opt*	*#opt BDP*	*BDP promedio CPU*	*BDP max CPU*
1	8	0	0.0003	0.0003
4	67	0	0.0011	0.0012
16	171	7	0.0051	0.0062
64	216	39	0.0263	0.0425
256	225	187	0.0790	0.2738
512	225	225	0.0887	0.4194

Instancias de estudio en la Planta de motores Nissan-BCN

En la planta de *Powertrain* de Nissan en Barcelona disponemos de una línea con 21 estaciones de trabajo ($m_1, ..., m_{21}$) en la que se montan 9 tipos de motores ($p_1, ..., p_9$) agrupados en 3 familias (4x4, furgonetas y camiones). Las 380 tareas que, aproximadamente, requiere el montaje de un motor, las agrupamos en 140 operaciones y se resolvió un problema *TSALB-1* (ver (Bautista and Pereira, 2007)) considerando unos tiempos de proceso promedio (idéntica presencia de los 9 tipos de motores en el mix de producción), un tiempo de ciclo $c = 180$ segundos y un área disponible por estación $A = 4m$, dando lugar a las 21 estaciones mencionadas (ver detalles en http://www.nissanchair.com/TSALBP). Tras la asignación de las operaciones a las estaciones, hemos calculado los tiempos de proceso de cada tipo de motor en cada estación, $p_{i,k}(i = 1, ..., 9; k = 1, ..., 21)$, y hemos supuesto $b_k = 1$ $(k = 1, ..., 21)$, dando como resultado los datos recogidos en la Tabla 2.7.

Tabla 2.7: Tiempos de proceso, $p_{i,k}$, de las operaciones sobre los 9 tipos de motores ($p_1, ..., p_9$) en las 21 estaciones ($m_1, ..., m_{21}$)

i	Tiempos de proceso para los productos en las estaciones																				
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}	m_{17}	m_{18}	m_{19}	m_{20}	m_{21}
p_1	104	103	165	166	111	126	97	100	179	178	161	96	99	147	163	163	173	176	162	164	177
p_2	100	103	156	175	114	121	96	97	174	172	152	106	101	155	152	185	179	167	150	161	161
p_3	97	105	164	172	114	122	96	95	173	172	168	105	102	142	156	183	178	181	152	157	154
p_4	92	107	161	167	115	124	93	106	178	177	167	97	101	154	152	178	169	180	152	159	168
p_5	100	101	148	168	117	127	96	94	178	178	167	101	99	146	153	169	173	172	160	162	172
p_6	94	108	156	167	117	130	89	102	171	177	166	100	101	143	152	173	178	173	151	160	170
p_7	103	106	154	168	115	120	94	103	177	175	172	96	96	154	154	172	174	173	155	162	167
p_8	109	102	164	156	111	121	101	102	171	173	157	104	102	153	156	182	175	168	148	158	149
p_9	101	110	155	173	111	134	92	100	174	175	177	96	99	155	156	171	175	184	167	157	169
l_k	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195	195
b_k	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

En este contexto, hemos considerado 46 ejemplares, asociados a planes de demanda, agrupados

en dos bloques (ver Tabla 2.8). El *Bloque I* presenta instancias (de 1 a 23) con planes de demanda, para los 9 tipos de motores, de 270 unidades ($T = 270$); mientras que la demanda total para las instancias del *Bloque II* (de 24 a 46) es $T = 540$.

Tabla 2.8: Instancias NISSAN-9ENG. Planes de demanda

Familia	i	*Bloque I*																					
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*
4x4	p_1	30	30	10	40	40	50	20	20	70	10	10	24	37	37	24	30	30	60	10	20	60	20	10
	p_2	30	30	10	40	40	50	20	20	70	10	10	23	37	37	23	30	30	60	10	20	60	20	10
	p_3	30	30	10	40	40	50	20	20	70	10	10	23	36	36	23	30	30	60	10	20	60	20	10
camionetas	p_4	30	45	60	15	60	30	75	30	15	105	15	45	35	45	55	35	55	30	90	15	15	90	30
	p_5	30	45	60	15	60	30	75	30	15	105	15	45	35	45	55	35	55	30	90	15	15	90	30
camiones	p_6	30	23	30	30	8	15	15	38	8	8	53	28	23	18	23	28	18	8	15	45	15	8	45
	p_7	30	23	30	30	8	15	15	38	8	8	53	28	23	18	23	28	18	8	15	45	15	8	45
	p_8	30	22	30	30	7	15	15	37	7	7	52	27	22	17	22	27	17	7	15	45	15	7	45
	p_9	30	22	30	30	7	15	15	37	7	7	52	27	22	17	22	27	17	7	15	45	15	7	45
		Bloque II																					
		24	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*	*44*	*45*	*46*
4x4	p_1	60	60	20	80	80	100	40	40	140	20	20	47	74	74	47	60	60	120	20	40	120	40	20
	p_2	60	60	20	80	80	100	40	40	140	20	20	47	73	73	47	60	60	120	20	40	120	40	20
	p_3	60	60	20	80	80	100	40	40	140	20	20	46	73	73	46	60	60	120	20	40	120	40	20
camionetas	p_4	60	90	120	30	120	60	150	60	30	210	30	90	70	90	110	70	110	60	180	30	30	180	60
	p_5	60	90	120	30	120	60	150	60	30	210	30	90	70	90	110	70	110	60	180	30	30	180	60
camiones	p_6	60	45	60	60	15	30	30	75	15	15	105	55	45	35	45	55	35	15	30	90	30	15	90
	p_7	60	45	60	60	15	30	30	75	15	15	105	55	45	35	45	55	35	15	30	90	30	15	90
	p_8	60	45	60	60	15	30	30	75	15	15	105	55	45	35	45	55	35	15	30	90	30	15	90
	p_9	60	45	60	60	15	30	30	75	15	15	105	55	45	35	45	55	35	15	30	90	30	15	90

Para el experimento, hemos considerado un tiempo de ciclo efectivo $c = 175s$, y una ventana temporal idéntica para todas las estaciones $l_k = 195s$ ($k = 1, ..., 21$), lo que supone una holgura superior al 10 % sobre el ciclo. Estos datos indican que las instancias con $T = 270$ están vinculadas a una jornada laboral de $13{,}125h$ (en dos turnos), mientras que las instancias con $T = 540$ representan 2 jornadas laborales.

Hemos recogido en la Tabla 2.9 las mejores soluciones para las 46 instancias del conjunto NISSAN-9ENG. Dichas soluciones se han obtenido a partir de dos vías:

Tabla 2.9: Resultados ofrecidos por *CPLEX (3600s)* y *BDP* ($H = (1, 10, 50, 100, 250, 500, 750, 1000)$) para las 46 instancias NISSAN-9ENG. Se recogen los valores de W, el tiempo de CPU y las cotas inferiores ofrecidas por ambos procedimientos.

	Instancia	Cota inferior		CPLEX 3600s	BDP H=1		BDP H=10		BDP H=50		BDP H=100		BDP H=250		BDP H=500		BDP H=750		BDP H=1000	
		CPLEX	BDP	W	W	CPU	W	CPU	W	CPU	W	CPU	W	CPU	W	CPU	W	CPU	W	CPU
	1	50	52	2550	760	0.15	343	1.75	240	14.14	189	42.00	179	202.70	179	716.98	**166**	1576.33	166	2659.14
	2	242	244	1073	572	0.15	572	1.72	516	13.72	470	39.85	470	193.17	**464**	687.43	464	1496.56	464	2472.36
	3	421	423	2267	751	0.15	502	1.71	470	13.65	459	38.97	454	188.57	**432**	648.60	432	1285.03	432	2190.19
	4	235	240	2675	934	0.15	585	1.71	473	13.48	473	38.40	443	185.73	443	659.92	443	1404.98	**440**	2460.43
	5	557	558	1587	1031	0.15	990	1.65	960	11.99	922	31.81	922	146.36	902	446.73	**897**	1120.36	897	1646.79
	6	285	296	1459	1205	0.15	755	1.70	697	12.93	679	37.12	675	173.02	667	590.21	667	1289.57	**663**	2153.41
	7	721	723	1403	893	0.15	893	1.72	873	13.22	847	36.12	826	170.66	**823**	571.93	823	1332.27	823	2319.87
	8	72	72	2322	570	0.15	232	1.77	177	14.51	177	42.71	146	213.94	146	769.60	**129**	1659.06	129	2716.60
	9	657	662	2084	1507	0.15	1229	1.62	1172	11.47	1170	32.65	1166	141.66	1150	448.67	1150	996.32	**1149**	1498.92
	10	1210	1211	1321	1279	0.15	1279	1.62	1271	11.38	1263	31.64	1263	143.73	1251	432.84	1251	869.04	**1249**	1845.27
	11	43	43	1996	806	0.15	94	1.80	61	14.21	60	40.32	60	192.07	52	716.13	**50**	1270.12	50	2059.97
Bloque I	12	227	230	2490	540	0.15	461	1.75	419	14.06	410	41.70	402	197.11	402	701.36	373	1595.74	**369**	2757.74
	13	164	165	2722	834	0.15	474	1.73	437	13.52	431	38.50	385	195.16	385	651.54	385	1443.35	**379**	2554.69
	14	290	291	2787	745	0.15	671	1.71	655	13.19	624	37.13	616	177.15	**578**	625.63	578	1329.50	578	2372.56
	15	393	395	2555	694	0.15	652	1.73	613	13.90	612	40.66	**553**	198.40	553	706.83	553	1497.23	553	2617.76
	16	96	99	2577	721	0.15	394	1.73	286	14.11	286	40.58	265	197.53	245	724.88	**223**	1570.60	223	2578.07
	17	409	411	2707	695	0.15	695	1.70	672	13.22	672	37.97	665	180.51	**640**	643.80	640	1384.36	640	2332.40
	18	456	467	1988	1380	0.15	1053	1.63	986	11.82	986	31.70	981	140.71	968	458.02	**962**	1001.94	962	1670.65
	19	947	948	1438	1022	0.15	1012	1.66	982	11.54	982	33.55	982	140.36	**980**	482.56	980	1005.56	980	1824.70
	20	50	50	2230	801	0.15	168	1.79	138	14.65	138	43.23	**104**	218.82	104	754.48	104	1609.10	104	2800.60
	21	480	491	3120	1317	0.15	964	1.67	877	12.63	877	34.32	869	163.68	**854**	575.48	854	1148.40	854	2031.87
	22	984	986	1200	1164	0.15	1164	1.65	1129	11.97	1125	31.97	1125	142.53	1125	467.70	1125	1022.01	**1104**	1813.15
	23	100	100	2105	659	0.15	137	1.77	129	14.07	129	40.74	120	192.88	118	714.10	118	1439.44	**107**	2630.03
	24	140	142	5100	1524	0.97	1031	10.20	517	61.77	480	151.33	453	578.37	**390**	1843.82	390	3658.71	390	6107.49
	25	525	528	5295	1365	0.98	1153	10.18	1089	61.03	1089	146.71	1036	544.76	1036	1657.35	1036	3368.61	**1026**	5570.80
	26	900	903	1875	1563	0.97	1105	10.15	1082	60.58	1008	146.45	1008	547.61	971	1711.14	967	3469.07	**932**	5584.50
	27	510	515	5271	1884	0.98	1234	10.15	1003	60.75	997	144.75	949	542.06	942	1664.14	942	3341.58	**941**	5634.32
	28	1195	1199	-	2166	0.98	2049	10.01	1986	57.21	1971	132.38	1956	469.08	1940	1325.13	1918	2730.52	**1915**	4132.01
	29	630	641	-	2480	0.97	1587	10.10	1475	59.47	1468	141.09	1465	512.78	1441	1546.83	**1421**	3103.84	1421	5181.39
	30	1500	1503	2672	1899	0.98	1899	10.15	1800	60.36	1798	144.47	1759	531.10	1757	1610.47	**1742**	3024.21	1742	5433.75
	31	160	160	1545	1266	0.98	537	10.26	381	62.30	378	153.43	321	594.05	299	1885.86	**284**	3847.76	284	6211.87
	32	1365	1376	-	3125	0.96	2527	9.96	2475	57.72	2407	133.89	2407	462.41	2392	1357.99	**2391**	2606.97	2391	4298.42
	33	2476	2478	2704	2604	0.97	2604	9.97	2599	56.37	2582	127.96	2582	424.88	2565	1281.61	**2563**	2441.75	2563	3990.37
	34	120	120	1357	1612	0.97	234	10.33	167	62.19	144	150.61	144	594.16	139	1871.63	**137**	3850.44	137	5912.36
Bloque II	35	476	479	4931	1125	0.97	939	10.20	884	61.87	829	151.54	817	573.98	817	1757.61	817	3608.43	**777**	6155.86
	36	382	385	5458	1868	0.97	1244	10.15	962	60.60	940	145.71	908	548.51	908	1638.33	**896**	3373.82	896	5540.99
	37	650	654	-	1676	0.97	1478	10.12	1347	60.51	1347	142.46	1347	518.74	1346	1580.17	**1299**	3237.03	1299	5373.71
	38	823	826	5125	1429	0.97	1388	10.18	1322	61.43	1269	148.30	1248	556.21	1219	1719.23	**1159**	3541.42	1159	5844.94
	39	245	248	5092	1532	0.97	910	10.20	666	61.79	596	150.43	596	557.91	589	1724.68	**545**	3623.51	545	5986.08
	40	875	878	2492	1392	0.97	1392	10.00	1392	60.27	1392	144.47	1392	525.51	1392	1634.58	1392	3223.35	**1380**	5402.38
	41	975	986	-	2855	0.96	2142	9.98	2077	57.66	2074	133.59	2053	460.19	2042	1327.12	**2036**	2691.20	2036	4370.09
	42	1951	1953	2741	2214	0.98	2110	10.10	2042	58.26	2026	134.41	2026	449.38	2026	1340.81	**2011**	2671.99	2011	4261.27
	43	130	130	1456	1549	0.97	454	10.27	307	63.60	256	155.83	256	600.31	256	1911.06	**234**	3942.54	234	6490.56
	44	1021	1031	6191	2732	0.96	1974	10.02	1864	58.63	1814	139.31	1814	498.16	1808	1482.77	1807	2971.74	**1786**	4964.17
	45	2026	2028	3199	2376	0.97	2376	10.02	2336	57.71	2312	132.80	2312	455.27	2312	1294.34	**2275**	2585.56	2275	4145.27
	46	220	220	4159	1360	0.97	367	10.26	287	62.38	280	149.34	253	551.44	250	1753.98	**233**	3446.11	233	5789.51

1. Debido a la falta de soluciones de referencia para las instancias relacionadas con el caso de estudio, se ha empleado el Solver CPLEX v11.0 (licencia 1 solo procesador) corriendo en un ordenador MacPro con CPU Intel Xeon 3 GHz y 2 Gb de RAM en sistema operativo Windows XP, limitando el tiempo de CPU primero, a $3600s$, y en caso de no obtener solución hemos relanzado la ejecución de CPLEX con un tiempo limitado a $7200s$. Los resultados para W se recogen en la columna *CPLEX 3600s* de la Tabla 2.9.

2. *BDP* programada en gcc v. 4.01 corriendo en un ordenador Macintosh iMac con un procesador Intel Core 2 Duo, 2.33 Ghz. y 3 Gb de memoria RAM, en idénticas condiciones al empleado en el apartado 2.3.7. Las 46 instancias se han resuelto empleando 8 anchos de ventana crecientes (1, 10, 50, 100, 250, 500, 750 y 1000); hemos considerado un número máximo de transiciones desde cada vértice igual a 9 (número de tipos de motores); y hemos empleado, para cada ancho de ventana, como valor de una solución inicial, Z_0, el valor de la solución obtenida por *BDP* con el ancho de ventana anterior, excepto para el caso $H = 1$, donde se fijo $H = 1$ a ∞. Los resultados de W y tiempos de CPU se recogen, también, en la Tabla 2.9.

A la vista de la Tabla 2.9, podemos hacer las siguientes observaciones:

- CPLEX no obtuvo solución en 5 instancias (28, 29, 32, 37 y 41) durante 2 horas de ejecución.
- *BDP* con $H = 1$ obtuvo mejor solución que CPLEX en 44 de 46 instancias. En las instancias 34 y 43 CPLEX obtuvo soluciones ligeramente mejores que *BDP* $H = 1$.
- *BDP H = 10* superó a CPLEX en todos los casos empleando unos tiempos de CPU inferiores a 2 segundos, para las instancias del *Bloque I*, e inferiores a 11 segundos para las del *Bloque II*.
- La mejora promedio sobre la sobrecarga W dada por *BDP* con $H = 1$ sobre CPLEX fue: $\triangle\bar{W} = \bar{W}_{CPLEX} - \bar{W}_{BDP-1} = 1411{,}01$.
- La mejora promedio sobre W dada por *BDP* con $H = 1000$ frente a *BDP* con $H = 1$ fue: $\triangle\bar{W} = \bar{W}_{BDP-1} - \bar{W}_{BDP-1000} = 495{,}39$, mientras que el tiempo de CPU promedio pasó de $0{,}56s$ a $3791{,}07s$.
- Los valores de la sobrecarga promedio $\bar{W}$ obtenidos por *BDP* decrecen, con saturación, a medida que aumentamos el ancho de ventana, desde $\bar{W}_{BDP-1} = 1401{,}65$ hasta $\bar{W}_{BDP-1000} = 906{,}26$.
- En todos los ejemplares, *BDP* siempre encuentra una cota inferior de W igual o mejor que la ofrecida por CPLEX.
- No podemos garantizar haber obtenido una solución óptima en alguna de las 46 instancias.

2.3.8. Conclusiones del uso de la *BDP* para el problema *MMSP-W* con estaciones en serie considerando instantes de interrupción restringidos

A partir del apartado 2.3 de este capítulo, se han presentado dos modelos $M3$ y $M4$, sobre secuenciación de productos para minimizar la sobrecarga de trabajo en una línea de montaje con vínculos entre sus estaciones. Dichos modelos pueden verse como una extensión de los modelos propuestos por (Yano and Rachamadugu, 1991) y (Scholl et al., 1998).
Además, a través de un ejemplo, se ha presentado la utilidad que dichos modelos pueden tener cuando se dispone de un sistema de control de producción que puede informar, en cada momento, a los procesadores de las estaciones, sobre los instantes precisos en que se debe interrumpir o intensificar el trabajo sobre una operación; por lo que no siempre se podrán llevar a la práctica las soluciones ofrecidas por dichos modelos.
Debido a este caso, se ha propuesto una extensión sobre $M3$ y $M4$, dando lugar a los modelos $M5$ y $M6$, que incorporan reglas sobre la interrupción de las operaciones con el propósito de hacer más autónomas las estaciones y así facilitar la labor a control de producción.
Posteriormente, se ha propuesto un procedimiento basado en la programación dinámica acotada para resolver el problema de secuenciación minimizando la sobrecarga y considerando vínculos entre estaciones y reglas de interrupción de operaciones.
Para contrastar la calidad del procedimiento, se ha realizado una experiencia computacional con 225 ejemplares de la literatura utilizando el solver CPLEX, por un lado, y la implementación *BDP*, por otro, obteniendo todos los óptimos en unos tiempos de CPU promedio (por instancia) de $4347{,}76s$ para CPLEX y $0{,}0887s$ para *BDP*. En este experimento queda claro el predominio de la implementación *BDP* sobre CPLEX.
Una vez contrastada la calidad, se ha aplicado el procedimiento *BDP* a un caso de estudio relacionado con la línea de Powertrain de Nissan en Barcelona, a través de 23 instancias con $T = 270$ (una jornada de dos turnos) y otras 23 con $T = 540$ (dos jornadas), representando cada instancia un posible escenario. Para la obtención de soluciones, empleamos de nuevo el solver CPLEX y la implementación *BDP*, y observamos un claro predominio de *BDP* sobre CPLEX, tanto en soluciones como en cotas inferiores, sin poder garantizar que dichas soluciones son óptimas. Hablando de tiempos de CPU, de nuevo *BDP* fue mucho más rápida que CPLEX, como se puede observar en el caso de *BDP* con $H = 10$, donde se obtuvieron mejores valores para W en todas las instancias cuando se comparo con CPLEX, en un tiempo mucho menor: menos de 2 segundos para el *Bloque I* y menos de 11 segundos para el *Bloque II*.

2.4. *MMSP-W* con estaciones en serie considerando libre interrupción de operaciones

Hasta este momento se ha considerado vínculos entre estaciones en serie y se han propuesto modelos tanto para la libre interrupción de operaciones (modelos $M3$ y $M4$) como para los instantes de interrupción restringidos al fin del ciclo y la ventana temporal (modelos $M5$ y $M6$).
Sin embargo, a diferencia del apartado 2.3, donde se han propuesto modelos y un procedimiento de resolución para el problema, *MMSP-W* cuando no disponemos de un sistema de control preciso para indicar a las diferentes estaciones de trabajo el instante en que deben interrumpir su operación, en este apartado vamos a considerar que si disponemos de dicho sistema, ya que por ejemplo existen líneas de producción altamente automatizadas donde puede ser de interés la resolución del *MMSP-W* con estaciones en serie y libre interrupción de operaciones.

2.4.1. Modelo para el *MMSP-W* con estaciones en serie considerando libre interrupción de operaciones

Para esta variante, disponemos de los modelos $M3$ y $M4$ formulados previamente en el apartado 2.3.1. En este caso, vamos a usar el modelo $M4$, debido a que pruebas anteriores a dicha tesis demostraron que el solver CPLEX resuelve el modelo $M4$ en un tiempo de CPU inferior al del modelo equivalente $M3$, y como se ha demostrado en el Teorema 1, ambas funciones objetivo son equivalentes.
El modelo $M4$ presentado en el apartado 2.3.1 tiene como función objetivo minimizar la sobrecarga total (2.29) y está sujeto a las restricciones (2.30) - (2.39). Dichas restricciones ya consideran el vínculo entre estaciones y el libre instante de interrupción, tal y como se puede observar en la Figura 2.6.

2.4.2. *BDP* para el problema

De nuevo, vamos a emplear un esquema basado en programación dinámica acotada para resolver el problema. En este apartado vamos a describir los elementos necesarios para el algoritmo *BDP*.

Cotas globales y parciales

Como en el problema anterior, debemos definir cotas para el problema con el objetivo de disponer de una guía para el procedimiento. En este caso, debido a que los instantes de interrupción son libres, no es trivial obtener una cota, ya que el proceso de la pieza se puede interrumpir en cualquier instante comprendido entre el tiempo de ciclo y la ventana temporal. Por dicho motivo, es necesario el uso de un modelo lineal para determinar la sobrecarga de la parte secuenciada. Sin

embargo, para obtener una cota para la sobrecarga del complemento no secuenciado, es posible utilizar cotas similares a las descritas anteriormente o, como es el caso de este procedimiento *BDP*, utilizar el mismo programa lineal usado para determinar la sobrecarga de la parte ya secuenciada. Dado un vértice de la etapa t, alcanzado a través de la secuencia parcial $\pi(t) = \{\pi_1, \pi_2, ...\pi_t\}$, la cota global para W y una cota parcial para el complemento $R(\pi(t))$ asociado a la secuencia o segmento $\pi(t)$ se puede determinar de acuerdo con el esquema presentado en la Figura 2.10.

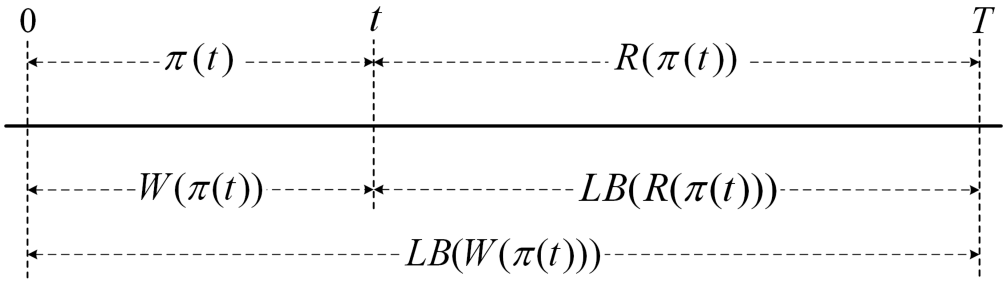

Figura 2.10: Esquema de acotación para una secuencia parcial $\pi(t)$

Para obtener las cotas de las sobrecargas asociadas a $\pi(t)$ y $R(\pi(t))$, utilizaremos el modelo matemático $M4$ al cual se le imponen las siguientes condiciones:

- Los valores de las variables $x_{i,\tau}$ $(i = 1, ..., |I|; \tau = 1, ..., t)$ se fijan en consonancia con la secuencia parcial $\pi(t)$:

$$x_{i,\tau} = \begin{cases} 1 & \text{Si } \pi_\tau = i, \\ 0 & \text{Si } \pi_\tau \neq i. \end{cases} \Leftrightarrow \begin{cases} x_{\pi_\tau,\tau} = 1 \\ x_{i,\tau} = 0, \quad \text{Si } i \neq \pi_\tau \end{cases} \tag{2.87}$$

- La condición de binariedad es relajada para las variables $x_{i,\tau} (i = 1, ..., |I|; \tau = t + 1, ..., T)$:

$$0 \leqslant x_{i,\tau} \leqslant 1 \qquad i = 1, ..., |I|; \tau = t + 1, ..., T \tag{2.88}$$

El resultado es el siguiente programa matemático, al que denominaremos LB_M4:

$$Min \quad LB(W(\pi(t))) = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \tag{2.89}$$

Sujeto a:

$$(2.30) - (2.37) \; y \; (2.39) \; de \; M4 \tag{2.90}$$

$$x_{\pi_\tau,\tau} = 1 \qquad \forall \tau = 1, ..., t \tag{2.91}$$

$$0 \leqslant x_{i,\tau} \leqslant 1 \qquad \forall i = 1, ..., |I|; \forall \tau = t + 1, ..., T \tag{2.92}$$

El modelo LB_M4 provee de una cota global para el valor de la sobrecarga asociado al segmento $\pi(t)$ $(W(\pi(t)))$ y una cota de la sobrecarga asociada al complemento $R(\pi(t))$ $(LB(R(\pi(t))))$. Estos

valores se pueden calcular tal y como sigue:

$$W(\pi(t)) = \sum_{k=1}^{|K|} \left(b_k \sum_{\tau=1}^{t} w_{k,\tau} \right) \tag{2.93}$$

$$LB(R(\pi(t))) = \sum_{k=1}^{|K|} \left(b_k \sum_{\tau=t+1}^{T} w_{k,\tau} \right) \tag{2.94}$$

Los instantes de finalización relativos ($\hat{e}_{k_t}$) de la última operación de la secuencia parcial $\pi(t)$, en cada estación de trabajo, también se puede obtener de LB_M4 como sigue:

$$\hat{e}_{k,t} = \hat{s}_{k,t} + \rho_{k,t} - w_{k,t} \qquad \forall k = 1, ..., |K| \tag{2.95}$$

Grafo asociado al problema

De forma análoga al apartado 2.3.4, podemos de nuevo construir un grafo conectado sin bucles ni ciclos directos de $T+1$ etapas. El conjunto de vértices en el nivel t ($t = 0, ..., T$) lo notaremos como $J(t)$. Sea $J(t,j)$ ($j = 1, ..., |J(t)|$) un vértice del nivel t, éste queda definido por la tupla $(\vec{q}(t,j), \vec{e}(t,j), \pi(t,j), LB(W(\pi(t,j))))$, donde:

- $\vec{q}(t,j) = (q_1(t,j), ..., q_{|I|}(t,j))$ representa el vector de demanda satisfecha asociada al vértice.
- $\vec{e}(t,j) = (e_1(t,j), ..., e_{|K|}(t,j))$ representa el vector de instantes de finalización absolutos, en las estaciones, de las operaciones asociadas al vértice.
- $\pi(t,j)$ representa la secuencia de t unidades de productos asociados al vértice.
- $LB(W(\pi(t,j)))$ es la cota de la sobrecarga total, considerando la secuencia $\pi(t,j)$, obtenida a través del programa lineal LB_M4.

Además, el vértice $J(t,j)$ presenta las siguientes propiedades:

$$\sum_{i=1}^{|I|} q_i(t,j) = t \tag{2.96}$$

$$0 \leqslant q_i(t,j) \leqslant d_i \qquad i = 1, ..., |I| \tag{2.97}$$

$$e_k^c(t,j) = max\left\{(t+k-2)c + \hat{e}_{k,t}, (t+k-1)c\right\} \qquad k = 1, ..., |K| \tag{2.98}$$

siendo $\vec{e}^{\,c}(t,j)$ el vector de instantes de finalización corregidos en consonancia con el tiempo de ciclo.
En definitiva, un vértice $J(t,j)$ lo representaremos de la forma:

$$J(t,j) = \{(t,j), \vec{q}(t,j), \pi(t,j), LB(W(\pi(t,j))), \vec{e}(t,j), \vec{e}^{\,c}(t,j)\} \tag{2.99}$$

En el nivel 0 del grafo hay un sólo vértice $J(0)$. Inicialmente, podríamos pensar que en el nivel t, $J(t)$ contiene los vértices asociados a todas las subsecuencias que pueden construirse con t productos satisfaciendo las propiedades (2.96), (2.97) y (2.98), no obstante, es fácil reducir el cardinal que a priori puede presentar $J(t)$ estableciendo las siguientes relaciones de pseudo-dominancia: dadas las secuencias $\pi(t,j_1)$ y $\pi(t,j_2)$ asociadas a los vértices $J(t,j_1)$ y $J(t,j_2)$, entonces $\pi(t,j_1)$ pseudo-domina a $\pi(t,j_2)$ si:

$$\begin{aligned}\pi(t,j_1) \prec \pi(t,j_2) \Leftrightarrow [\vec{q}(t,j_1) = \vec{q}(t,j_2)] \wedge [LB(W(\pi(t,j_1))) \leqslant LB(W(\pi(t,j_2)))] \wedge \\ \wedge [W(\pi(t,j_1)) \leqslant W(\pi(t,j_2))] \wedge [\vec{e}^c(t,j_1) \leqslant \vec{e}^c(t,j_2)]\end{aligned} \tag{2.100}$$

La reducción de $J(T)$ a través de las pseudo-dominancias definidas en (2.100) no puede garantizar la optimalidad de las soluciones.

El uso de la *BDP*

Para esta nueva variante, de nuevo utilizamos el procedimiento basado en *BDP*, ya que esta tesis doctoral investiga en la aplicación sobre diferentes problemas de la literatura. De modo resumido, el algoritmo *BDP* consiste en (ver detalles en el apartado 2.3.6):

Algoritmo 4 *Esquema BDP para el problema MMSP-W con instantes de interrupción libres*

Entrada: $T, |I|, |K|, d_i (i = 1, ..., |I|), p_{i,k} (i = 1, ..., |I|; k = 1, ..., |K|), l_k, b_k (\forall k), c, Z_0, H$
Salida: Lista de secuencias obtenidas con *BDP*

```
Inicialización: t = 0; LBZ_min = ∞
mientras t < T hacer
    t ← t + 1
    mientras lista de vértices consolidados en la etapa t − 1 no vacía hacer
        Seleccionar_vértice(t)
        Desarrollar_vértice(t)
        Filtrar_vértices(Z_0, H, LBZ_min)
    fin mientras
    Finalizar_etapa()
fin mientras
```

En el procedimiento aparecen las siguientes funciones:

- Seleccionar_vértice(t): selecciona, siguiendo un orden no decreciente de los valores $LB(W(\pi(t-1,j)))$, uno de los vértices consolidados en la etapa $t-1$.
- Desarrollar_vértice(t): desarrolla el vértice seleccionado en el paso anterior añadiendo un nuevo producto con demanda pendiente. Se calcula la cota $LB(W(\pi(t,j)))$ usando el modelo matemático LB_M4.

- Filtrar_vértices(Z_0, H, LBZ_{min}): escoge, de todos los vértices desarrollados en el paso anterior, un número máximo H de los vértices más prometedores (de acuerdo con el menor valor de la cota inferior $LB(W(\pi(t,j)))$). Además, se eliminan los vértices cuya cota inferior sea igual o superior a una solución conocida (Z_0), se rechazan los vértices generados pseudo-dominados por aquellos vértices ya escogidos previamente y se desplazan los vértices escogidos previamente pseudo-dominados por los nuevos vértices.
- Finalizar_etapa(): consolida los vértices más prometedores de la etapa t (como máximo un número H de éstos).

2.4.3. Experiencia computacional

Para probar el rendimiento y la calidad del procedimiento *BDP*, en este caso se contrastó los resultados obtenidos por dicho procedimiento con los resultados obtenidos por $M4$, usando las 225 instancias presentadas en el apartado 2.3.7.
Las soluciones de referencia para el modelo $M4$ se obtuvieron usando el solver CPLEX v11.0 (licencia de un único procesador) y fueron comparadas con las soluciones ofrecidas por el procedimiento *BDP* propuesto, bajo las siguientes condiciones:

1. Procedimiento *BDP* programado en C++, usando gcc v4.2.1, en un ordenador Apple Macintosh iMac con un procesador Intel Core i7 a 2.93 GHz y 8 Gb de RAM usando MAC OS X 10.6.7 (sin usar ningún tipo de código en paralelo; por lo tanto, el procesador se puede considerar con un único procesador a 2.93 GHz).
2. El procedimiento *BDP* usa 6 anchos de ventana (H), con valores 1,4, 16, 64, 256 y 1024.
3. Como solución inicial (Z_0) para cada ancho de ventana se utilizó la solución obtenida con *BDP* en el ancho anterior, excepto en el caso $H = 1$ en el que Z_0 se estableció como ∞.
4. Para calcular las cotas inferiores, $LB(W(\pi(t,j)))$, de la sobrecarga asociada a cada vértice en el procedimiento *BDP*, se utilizó el solver Gurobi v4.5.0, resolviendo el programa lineal asociado a LB_M4. El motivo del cambio de solver es que se investigó el uso de las librerías que CPLEX y Gurobi ofrecen para C++, resultando mucho más eficiente las herramientas ofrecidas por Gurobi. A partir de este punto, se desestimo el uso de CPLEX para el resto de esta tesis, así como se volvió a resolver el modelo $M4$. Sin embargo, dado que los resultados para $M4$ ofrecidos por ambos solvers son iguales en cuanto a sobrecarga y únicamente difieren en tiempo de ejecución (aunque siendo muy semejantes pero ligeramente mejores usando Gurobi), se ha optado por mantener en este apartado las soluciones de CPLEX y en posteriores apartados, se utilizarán las soluciones obtenidas con Gurobi para su contraste.

Las Tablas 2.10 a 2.14 muestran los resultados obtenidos con *BDP* y CPLEX para las 225 instancias. En cada una de las tablas, cada una de ellas correspondiente a una estructura, se muestran:

1. La columna '*Ins.*' corresponde al código de la instancia (programa de producción/estructura).
2. El valor de la sobrecarga total W de la solución óptima obtenido por CPLEX ('*OPT CPLEX*') resolviendo el modelo $M4$.
3. Los valores de las sobrecargas W obtenidas por el algoritmo *BDP* utilizando los diferentes anchos de ventana H ($H = 1, ..., 1024$).
4. El mejor valor de W para cada instancia obtenido por *BDP* ('*Mejor BDP*').
5. El mejor valor para cada instancia de la desviación porcentual relativa (*RPD*) obtenido ('*Mejor RPD*'), calculado como $RPD = ((Mejor\ BDP - Opt\ CPLEX)/Opt\ CPLEX) * 100$.

Tabla 2.10: Resultados para la estructura 1, para todos los programas de producción

Ins.	*Opt*	*H=1*	*H=4*	*H=16*	*H=64*	*H=256*	*H=1024*	*Mejor*	*Mejor*
	CPLEX	*W*	*W*	*W*	*W*	*W*	*W*	*BDP*	*RPD*
1/1	40	46	42	40	-	-	-	40	0.00
2/1	100	107	102	100	-	-	-	100	0.00
3/1	81	88	86	81	-	-	-	81	0.00
4/1	143	143	-	-	-	-	-	143	0.00
5/1	46	49	49	46	-	-	-	46	0.00
6/1	46	49	47	47	46	-	-	46	0.00
7/1	14	17	16	14	-	-	-	14	0.00
8/1	61	63	63	61	-	-	-	61	0.00
9/1	81	87	81	-	-	-	-	81	0.00
10/1	68	76	69	68	-	-	-	68	0.00
11/1	37	48	43	40	39	37	-	37	0.00
12/1	37	48	44	38	37	-	-	37	0.00
13/1	29	38	34	30	30	29	-	29	0.00
14/1	45	50	48	47	46	45	-	45	0.00
15/1	40	51	42	41	41	40	-	40	0.00
16/1	32	40	36	34	34	32	-	32	0.00
17/1	37	48	42	39	38	37	-	37	0.00
18/1	45	55	53	47	46	45	-	45	0.00
19/1	33	38	35	35	33	-	-	33	0.00
20/1	31	38	31	-	-	-	-	31	0.00
21/1	56	72	57	57	56	-	-	56	0.00
22/1	63	96	79	63	-	-	-	63	0.00
23/1	55	72	64	60	55	-	-	55	0.00
24/1	67	90	78	67	-	-	-	67	0.00
25/1	56	67	60	58	56	-	-	56	0.00
26/1	63	74	69	63	-	-	-	63	0.00
27/1	58	63	63	59	58	-	-	58	0.00
28/1	46	57	52	46	-	-	-	46	0.00
29/1	48	57	53	50	48	-	-	48	0.00
30/1	55	68	55	-	-	-	-	55	0.00
31/1	52	58	55	53	52	-	-	52	0.00
32/1	53	62	53	-	-	-	-	53	0.00
33/1	53	63	57	53	-	-	-	53	0.00
34/1	25	35	30	25	-	-	-	25	0.00
35/1	41	46	42	41	-	-	-	41	0.00
36/1	29	37	35	29	-	-	-	29	0.00
37/1	48	58	52	49	48	-	-	48	0.00
38/1	43	50	45	45	43	-	-	43	0.00
39/1	47	58	49	48	47	-	-	47	0.00
40/1	22	30	23	23	22	-	-	22	0.00
41/1	34	39	35	35	34	-	-	34	0.00
42/1	22	30	24	24	23	22	-	22	0.00
43/1	44	52	49	45	45	44	-	44	0.00
44/1	34	39	37	35	34	-	-	34	0.00
45/1	43	55	49	45	44	43	-	43	0.00

Tabla 2.11: Resultados para la estructura 2, para todos los programas de producción

Ins.	*Opt*	*H=1*	*H=4*	*H=16*	*H=64*	*H=256*	*H=1024*	*Mejor*	*Mejor*
	CPLEX	*W*	*W*	*W*	*W*	*W*	*W*	*BDP*	*RPD*
1/2	230	230	-	-	-	-	-	230	0.00
2/2	282	296	296	282	-	-	-	282	0.00
3/2	270	271	271	270	-	-	-	270	0.00
4/2	324	328	324	-	-	-	-	324	0.00
5/2	200	227	203	200	-	-	-	200	0.00
6/2	100	114	108	100	-	-	-	100	0.00
7/2	151	160	155	152	151	-	-	151	0.00
8/2	199	218	205	203	201	199	-	199	0.00
9/2	218	238	224	218	-	-	-	218	0.00
10/2	225	234	225	-	-	-	-	225	0.00
11/2	132	162	139	136	132	-	-	132	0.00
12/2	112	129	123	113	112	-	-	112	0.00
13/2	122	145	122	-	-	-	-	122	0.00
14/2	153	169	163	153	-	-	-	153	0.00
15/2	147	161	153	147	-	-	-	147	0.00
16/2	117	132	123	117	-	-	-	117	0.00
17/2	133	151	140	133	-	-	-	133	0.00
18/2	136	160	138	138	138	137	136	136	0.00
19/2	150	179	162	150	-	-	-	150	0.00
20/2	114	131	119	115	114	-	-	114	0.00
21/2	193	196	195	193	-	-	-	193	0.00
22/2	213	227	218	213	-	-	-	213	0.00
23/2	197	213	203	197	-	-	-	197	0.00
24/2	205	210	210	208	208	205	-	205	0.00
25/2	190	196	196	190	-	-	-	190	0.00
26/2	206	214	214	208	206	-	-	206	0.00
27/2	200	211	202	201	200	-	-	200	0.00
28/2	175	189	189	175	-	-	-	175	0.00
29/2	158	194	178	158	-	-	-	158	0.00
30/2	178	186	183	179	179	178	-	178	0.00
31/2	153	184	159	159	155	154	153	153	0.00
32/2	187	204	202	188	188	187	-	187	0.00
33/2	174	194	177	174	-	-	-	174	0.00
34/2	141	155	146	141	-	-	-	141	0.00
35/2	117	145	126	121	117	-	-	117	0.00
36/2	152	156	152	-	-	-	-	152	0.00
37/2	116	144	124	122	116	-	-	116	0.00
38/2	179	209	185	179	-	-	-	179	0.00
39/2	168	190	168	-	-	-	-	168	0.00
40/2	104	119	111	104	-	-	-	104	0.00
41/2	92	106	101	92	-	-	-	92	0.00
42/2	137	157	148	140	137	-	-	137	0.00
43/2	116	135	120	116	-	-	-	116	0.00
44/2	163	185	163	-	-	-	-	163	0.00
45/2	149	161	149	-	-	-	-	149	0.00

Tabla 2.12: Resultados para la estructura 3, para todos los programas de producción

Ins.	*Opt*	*H=1*	*H=4*	*H=16*	*H=64*	*H=256*	*H=1024*	*Mejor*	*Mejor*
	CPLEX	*W*	*W*	*W*	*W*	*W*	*W*	*BDP*	*RPD*
1/3	369	399	386	369	-	-	-	369	0.00
2/3	339	373	356	341	339	-	-	339	0.00
3/3	421	444	434	421	-	-	-	421	0.00
4/3	313	329	321	314	313	-	-	313	0.00
5/3	345	397	364	349	345	-	-	345	0.00
6/3	227	244	227	-	-	-	-	227	0.00
7/3	185	211	199	185	-	-	-	185	0.00
8/3	279	290	285	279	-	-	-	279	0.00
9/3	232	243	241	238	232	-	-	232	0.00
10/3	367	383	381	373	367	-	-	367	0.00
11/3	215	276	246	231	224	215	-	215	0.00
12/3	221	243	234	223	221	-	-	221	0.00
13/3	207	240	221	212	210	207	-	207	0.00
14/3	231	260	252	244	240	237	237	237	2.60
15/3	239	285	268	267	249	239	-	239	0.00
16/3	196	222	210	210	200	196	-	196	0.00
17/3	219	236	236	229	225	221	221	221	0.91
18/3	229	262	242	238	229	-	-	229	0.00
19/3	217	284	259	226	219	217	-	217	0.00
20/3	221	253	245	236	221	-	-	221	0.00
21/3	277	303	303	290	281	277	-	277	0.00
22/3	311	336	332	325	314	311	-	311	0.00
23/3	329	354	350	343	330	329	-	329	0.00
24/3	259	284	276	269	263	259	-	259	0.00
25/3	295	321	314	307	299	295	-	295	0.00
26/3	243	263	256	248	245	245	245	245	0.82
27/3	261	272	272	261	-	-	-	261	0.00
28/3	265	309	296	296	273	265	-	265	0.00
29/3	283	327	321	310	299	283	-	283	0.00
30/3	214	239	225	217	214	-	-	214	0.00
31/3	245	267	267	257	247	247	247	247	0.82
32/3	235	281	259	248	235	-	-	235	0.00
33/3	243	273	258	243	-	-	-	243	0.00
34/3	208	235	222	208	-	-	-	208	0.00
35/3	235	266	257	251	235	-	-	235	0.00
36/3	196	229	215	199	196	-	-	196	0.00
37/3	235	253	241	237	235	-	-	235	0.00
38/3	263	333	306	287	265	263	-	263	0.00
39/3	257	326	294	274	259	257	-	257	0.00
40/3	199	222	213	199	-	-	-	199	0.00
41/3	213	230	217	213	-	-	-	213	0.00
42/3	211	292	242	233	211	-	-	211	0.00
43/3	232	257	248	235	232	-	-	232	0.00
44/3	271	341	314	281	271	-	-	271	0.00
45/3	265	334	306	285	270	265	-	265	0.00

Tabla 2.13: Resultados para la estructura 4, para todos los programas de producción

Ins.	*Opt*	*H=1*	*H=4*	*H=16*	*H=64*	*H=256*	*H=1024*	*Mejor*	*Mejor*
	CPLEX	*W*	*W*	*W*	*W*	*W*	*W*	*BDP*	*RPD*
1/4	710	771	771	730	710	-	-	710	0.00
2/4	638	699	681	662	638	-	-	638	0.00
3/4	80	80	-	-	-	-	-	80	0.00
4/4	80	80	-	-	-	-	-	80	0.00
5/4	674	735	735	735	682	682	682	682	1.19
6/4	338	338	-	-	-	-	-	338	0.00
7/4	338	338	-	-	-	-	-	338	0.00
8/4	302	307	302	-	-	-	-	302	0.00
9/4	302	302	-	-	-	-	-	302	0.00
10/4	80	80	-	-	-	-	-	80	0.00
11/4	415	430	425	416	415	-	-	415	0.00
12/4	326	326	-	-	-	-	-	326	0.00
13/4	326	326	-	-	-	-	-	326	0.00
14/4	314	319	314	-	-	-	-	314	0.00
15/4	240	240	-	-	-	-	-	240	0.00
16/4	240	240	-	-	-	-	-	240	0.00
17/4	320	320	-	-	-	-	-	320	0.00
18/4	415	415	-	-	-	-	-	415	0.00
19/4	415	425	425	417	415	-	-	415	0.00
20/4	252	252	-	-	-	-	-	252	0.00
21/4	228	233	228	-	-	-	-	228	0.00
22/4	154	154	-	-	-	-	-	154	0.00
23/4	154	154	-	-	-	-	-	154	0.00
24/4	228	228	-	-	-	-	-	228	0.00
25/4	228	233	228	-	-	-	-	228	0.00
26/4	302	307	302	-	-	-	-	302	0.00
27/4	302	307	302	-	-	-	-	302	0.00
28/4	166	166	-	-	-	-	-	166	0.00
29/4	166	166	-	-	-	-	-	166	0.00
30/4	314	314	-	-	-	-	-	314	0.00
31/4	314	319	314	-	-	-	-	314	0.00
32/4	403	413	413	403	-	-	-	403	0.00
33/4	403	403	-	-	-	-	-	403	0.00
34/4	252	252	-	-	-	-	-	252	0.00
35/4	252	252	-	-	-	-	-	252	0.00
36/4	326	326	-	-	-	-	-	326	0.00
37/4	326	326	-	-	-	-	-	326	0.00
38/4	510	564	564	564	541	516	516	516	1.18
39/4	509	564	564	554	524	511	511	511	0.39
40/4	338	338	-	-	-	-	-	338	0.00
41/4	338	338	-	-	-	-	-	338	0.00
42/4	427	433	433	433	427	-	-	427	0.00
43/4	427	429	428	428	427	-	-	427	0.00
44/4	522	576	576	576	538	531	531	531	1.72
45/4	521	576	566	566	542	536	521	521	0.00

Tabla 2.14: Resultados para la estructura 5, para todos los programas de producción

Ins.	*Opt*	*H=1*	*H=4*	*H=16*	*H=64*	*H=256*	*H=1024*	*Mejor*	*Mejor*
	CPLEX	*W*	*W*	*W*	*W*	*W*	*W*	*BDP*	*RPD*
1/5	211	228	221	211	-	-	-	211	0.00
2/5	276	285	285	276	-	-	-	276	0.00
3/5	166	167	166	-	-	-	-	166	0.00
4/5	212	232	212	-	-	-	-	212	0.00
5/5	222	237	227	227	227	223	222	222	0.00
6/5	136	144	137	136	-	-	-	136	0.00
7/5	143	163	145	143	-	-	-	143	0.00
8/5	177	223	178	178	177	-	-	177	0.00
9/5	175	196	177	177	176	176	176	176	0.57
10/5	42	47	42	-	-	-	-	42	0.00
11/5	121	136	131	121	-	-	-	121	0.00
12/5	95	112	95	-	-	-	-	95	0.00
13/5	95	107	96	96	95	-	-	95	0.00
14/5	103	114	108	103	-	-	-	103	0.00
15/5	70	75	70	-	-	-	-	70	0.00
16/5	65	70	65	-	-	-	-	65	0.00
17/5	84	111	89	89	84	-	-	84	0.00
18/5	158	166	166	159	159	158	-	158	0.00
19/5	151	186	154	151	-	-	-	151	0.00
20/5	68	96	79	68	-	-	-	68	0.00
21/5	106	108	107	106	-	-	-	106	0.00
22/5	82	104	83	83	82	-	-	82	0.00
23/5	67	93	77	67	-	-	-	67	0.00
24/5	128	145	129	129	128	-	-	128	0.00
25/5	103	132	114	103	-	-	-	103	0.00
26/5	152	165	154	154	153	152	-	152	0.00
27/5	137	150	141	141	137	-	-	137	0.00
28/5	64	89	65	64	-	-	-	64	0.00
29/5	56	70	56	-	-	-	-	56	0.00
30/5	155	168	156	155	-	-	-	155	0.00
31/5	148	162	155	149	148	-	-	148	0.00
32/5	176	196	181	177	176	-	-	176	0.00
33/5	171	176	176	172	172	171	-	171	0.00
34/5	91	128	92	91	-	-	-	91	0.00
35/5	86	121	92	86	-	-	-	86	0.00
36/5	146	163	146	-	-	-	-	146	0.00
37/5	132	132	-	-	-	-	-	132	0.00
38/5	185	206	188	188	188	186	185	185	0.00
39/5	194	205	198	196	196	196	196	196	1.03
40/5	103	119	104	103	-	-	-	103	0.00
41/5	107	130	117	107	-	-	-	107	0.00
42/5	142	155	145	142	-	-	-	142	0.00
43/5	151	152	152	152	152	152	152	152	0.66
44/5	169	182	171	169	-	-	-	169	0.00
45/5	182	189	187	183	183	183	183	183	0.55

El procedimiento *BDP*, con las pseudo-dominancias definidas en (2.100), alcanza las soluciones óptimas en 213 de las 225 instancias. Más concretamente, las instancias para las que no se encontraron las soluciones óptimas están concentradas en las estructuras 3 (programas de producción 14, 17, 26 y 31), la estructura 4 (programas de producción 5, 38, 39 y 44) y la estructura 5 (programas de producción 9, 39, 43 y 45), en las Tablas 2.12, 2.13 y 2.14, respectivamente. Los valores promedio para la *RPD* de las instancias de estas estructuras son 0.11 %, 0.10 % y 0.06 %. El hecho de que en estas instancias no se encuentren las soluciones óptimas es debido al empleo de pseudo-dominancias, que como se ha comentado anteriormente, no garantizan alcanzar los óptimos. Sin embargo, debido a la reducción del espacio de búsqueda que se consigue, se ha decidido no cambiar su definición.

Para comparar la eficiencia computacional, en tiempos de CPU, del procedimiento *BDP* contra los procedimientos exactos del solver CPLEX, se ha resumido los tiempos de CPU necesarios para encontrar las mejores soluciones de las 225 instancias. La Tabla 2.15 muestra los tiempos mínimos, máximos y promedios de CPU, usados por ambos procedimientos para solucionar el conjunto de instancias, agrupados por programas de producción y estructuras de tiempos de proceso.

Tabla 2.15: Tiempos de CPU por bloques y estructuras

	M_4 - CPLEX			*BDP*		
	min CPU	*max CPU*	*Promedio*	*min CPU*	*max CPU*	*Promedio*
Bloque 1	0.0	1.5	0.4	0.0	1.3	0.5
Bloque 2	0.0	173.9	23.3	0.0	9.5	1.7
Bloque 3	5.1	2533.2	335.7	0.1	22.6	6.4
Bloque 4	6.6	111.5	35.2	0.1	10.5	3.7
Bloque 5	0.1	697.7	43.5	0.1	14.2	3.2
Estructura 1	0.0	301.3	46.9	0.0	16.6	4.3
Estructura 2	0.1	129.7	16.5	0.1	7.2	2.3
Estructura 3	0.1	180.3	32.2	0.3	22.6	5.9
Estructura 4	0.0	2533.2	288.5	0.0	7.2	1.2
Estructura 5	0.2	126.4	24.4	0.1	14.2	2.8

A la vista de la Tabla 2.15, en todos los casos (excepto el *Bloque 1*), los tiempos de CPU promedios usados por CPLEX son superiores a los empleados por el procedimiento *BDP*, siendo la *Estructura 4* el caso extremo, donde el procemiento *BDP* fue 239 más rápido que CPLEX, y en promedio 54 veces más rápido.

Posteriormente, se probó el algoritmo *BDP* con las instancias relacionadas con el caso de estudio de la planta de motores de Nissan Barcelona. Pese a los buenos resultados obtenidos para las 225 instancias, *BDP* daba resultados positivos pero el tiempo de computación se disparaba, debido a

la complejidad de resolver el problema lineal entero en cada uno de los vértices del grafo. Esto condujo a la incorporación de otro mecanismo de acotación para encontrar la sobrecarga de la parte ya secuenciada, como se muestra en apartados posteriores de esta tesis.

2.4.4. Conclusiones del uso de la *BDP* para el problema *MMSP-W* con estaciones en serie considerando libre interrupción de operaciones

En esta ocasión, el procedimiento *BDP* propuesto incorporó el uso de programación lineal para la obtención de cotas, debido a la dificultad de encontrar la sobrecarga de la parte ya secuenciada de una subsecuencia.
Los tiempos de CPU fueron competitivos al compararlos con CPLEX, en concreto, el procedimiento *BDP* fue en promedio 54 veces más rápido que CPLEX. En cuanto a la calidad de las soluciones, *BDP* obtuvo 213 de 225 óptimos para las instancias usadas. Para las 12 instancias que no se obtuvo la solución óptima, se encontraron soluciones, en promedio, a menos del 0.1 % del valor óptimo. Este fenómeno es debido al empleo de pseudo-dominancias para reducir el espacio de búsqueda. Para el caso de la planta de motores de Nissan, de momento el procedimiento es demasiado lento, ya que, por ejemplo, para resolver los ejemplares con 270 motores, con $H = 9$ necesita aproximadamente 1 hora de ejecución, siendo a nuestro parecer, demasiado lento todavía. Como se ha comentado en el apartado anterior, es necesario redefinir el empleo del programa lineal, pues es el motivo de la lentitud del procedimiento. Sin embargo, es imposible de momento desligar el cálculo de la cota de un programa lineal, con lo que en posteriores apartados exploramos ideas para reducir el espacio de búsqueda y el tiempo de proceso del algoritmo, a través de cambios en el esquema de acotación.

2.5. Incorporación de regularidad al mix de producción en el *MMSP-W*

Dado las dificultades encontradas en el apartado anterior, en este apartado se ha reducido el espacio de búsqueda a través de la incorporación al problema original de restricciones, así como la evolución del esquema de acotación para mejorar el rendimiento del procedimiento *BDP*.
Con el fin de restringir el espacio de búsqueda, es en nuestra opinión imprescindible que las nuevas restricciones sobre el problema estén acompañadas de necesidades o propiedades deseables en una línea de montaje. Es por eso que, observando la filosofía de producción de muchas empresas, la mayoría intentar gobernar su sistema productivo de la forma más eficiente posible. De hecho, este hecho es el eje principal de las filosofías *JIT* (Just-In-Time, Toyota) y *DS* (Douki Seisan, Nissan). En ambas filosofías, uno de los aspectos sobre los que hacen especial hincapié ambas es en la reducción de stocks al mínimo, ya sea de componentes o de productos finales, con el objetivo, entre varios, de reducir el coste de fabricación.
Así, si fijamos el stock como elemento relevante del sistema, es razonable incorporar al *MMSP-W* una nueva propiedad, que propicie que las secuencias de productos mixtos reduzcan al mínimo los niveles de stock de productos y/o de componentes. Para ello podemos limitar o minimizar la variación de las tasas de fabricación de los productos, como es el caso del denominado en la literatura *PRV*, o bien que la limitación o minimización corresponda a la variación de las tasas de consumo de componentes de los productos, tal como sucede en el *ORV*. En ambos casos, se trata de mantener dichas tasas constantes.
Es por dicho motivo, que al problema *MMSP-W* con vínculos entre estaciones y libre interrupción de operaciones se le incorporará propiedades vinculadas a las filosofías *JIT* y *DS*, en este caso, la regularidad del mix de producción, vinculada al problema *PRV*, reduciendo los stocks de productos finales a la salida de la línea de montaje. A dicho nuevo problema se le denominará como *MMSP-W-pmr* (Mixed Model Sequencing Problem with Work-overload minimization and Production Mix Restrictions) y puede ser resuelto a través de: (1) una nueva función multiobjetivo basada en la minimización de la sobrecarga y la variación del mix de producción; o (2) la incorporación al *MMSP-W* de nuevas restricciones para garantizar la regularidad del mix.
En el presente apartado, dicha regularidad se incorpora al problema a través del segundo enfoque, añadiendo al problema original nuevas restricciones.

2.5.1. Modelo para el *MMSP-W* incorporado regularidad al mix de producción

En este caso, usaremos como punto de partida el modelo $M4$ (ver apartado 2.3.1), al cual se le incorporan restricciones en el mix de producción. Para extender dicho modelo, son necesarios los siguientes parámetros adicionales:

Parámetros adicionales

$\dot{d}_i$	Tasa ideal de producción para el producto i. $\dot{d}_i = d_i/T$ $(i = 1, ..., \|I\|)$.

Bajo este nuevo enfoque podemos definir el siguiente modelo matemático, $M4\_pmr$:

$$Min \quad W = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right) \tag{2.101}$$

Sujeto a:

$$(2.30) - (2.39)\ de\ M4 \tag{2.102}$$

$$\sum_{\tau=1}^{t} x_{i,\tau} \geqslant \lfloor t \cdot \dot{d}_i \rfloor \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.103}$$

$$\sum_{\tau=1}^{t} x_{i,\tau} \leqslant \lceil t \cdot \dot{d}_i \rceil \qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.104}$$

Como podemos observar, el modelo $M4\_pmr$ está sujeto a las restricciones originales de $M4$, (2.30) a (2.39). A estas restricciones se ha añadido además las restricciones (2.103) y (2.104) que son aquellas que incorporan la preservación del mix del producción deseadas en el contexto *JIT* y *DS*. Adicionalmente, es posible medir la no-regularidad de una secuencia usando la siguiente función cuadrática:

$$\triangle_Q(X) = \sum_{t=1}^{T} \sum_{i=1}^{|I|} \left(X_{i,t} - t \cdot \dot{d}_i \right)^2 \tag{2.105}$$

donde $X_{i,t} = \sum_{\tau=1}^{t} x_{i,\tau}$ $(\forall i = 1, ..., |I|; \forall t = 1, ..., T)$ es la producción acumulada hasta el momento.

2.5.2. Un ejemplo

Partiendo del ejemplo presentado en el apartado 2.3.2, este se extiende añadiendo el hecho de que existen 2 procesadores asignados a la máquina m_2 (Tabla 2.16; el tiempo de ciclo sigue siendo $c = 4$ y $l_k = 6$):

Tabla 2.16: Número de procesadores homogéneos en cada estación y tiempos de proceso ($p_{i,k}$) para cada procesador requeridos por las unidades de producto, según tipo, en cada estación o módulo

	`A` $(d_A = 3)$	`B` $(d_B = 1)$	`C` $(d_C = 2)$	b_k
m_1	5	4	3	1
m_2	5	4	4	2
m_3	4	3	5	1
`Total`	19 $(V_0(A) = 57)$	15 $(V_0(B) = 15)$	16 $(V_0(C) = 32)$	$V_0 = 104$

La Figura 2.11 muestra un diagrama de Gantt de las soluciones óptimas ofrecidas por los modelos $M4$ (arriba en la figura) y $M4\_pmr$ (abajo en la figura). La secuencia de productos que conducen a la sobrecarga mínima total para $M4$ es $C-C-B-A-A-A$. El trabajo total completado es $V=101$, sobre un total $V_0=104$. La sobrecarga total, concentrada entre las estaciones m_1 y m_2, es $W=3$. La no-regularidad para este modelo es 9.05. En cambio, la secuencia que conduce a la menor sobrecarga total para $M4\_pmr$ es $C-A-B-A-C-A$ (debido a que la secuencia está afectada por las restricciones en el mix de producción). El trabajo total completado es de nuevo $V=101$, siendo la sobrecarga total, concentrada en m_1 y m_2, también $W=3$, mientras que en este caso la no-regularidad de la secuencia disminuyó hasta 2.05.

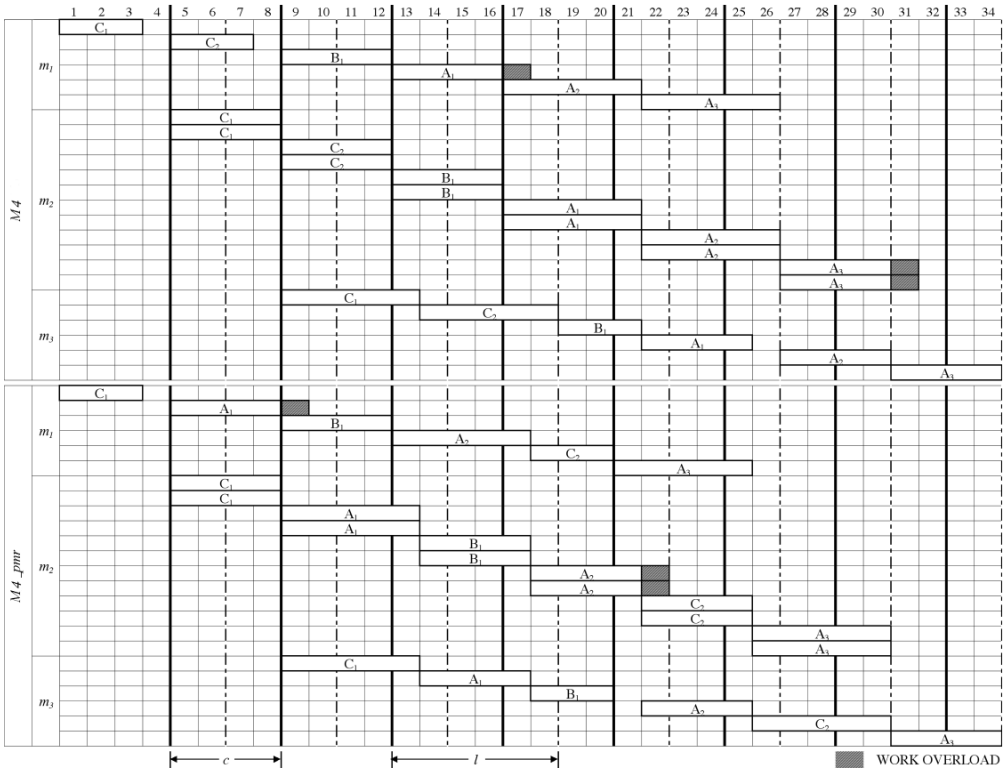

Figura 2.11: Solución óptima dada por $M4$ (arriba) y $M4\_pmr$ (abajo) para el Ejemplo

2.5.3. *BDP* para el *MMSP-W* con restricciones en el mix de producción (*pmr*)

Este apartado presenta los elementos básicos del procedimiento *BDP* aplicado al *MMSP-W* con estaciones en serie, libre instante de interrupción de las operaciones y restricciones en el mix de producción.

Grafo asociado al problema

De forma análoga al apartado 2.3.4, podemos de nuevo construir un grafo conectado sin bucles ni ciclos directos de $T+1$ etapas. El conjunto de vértices en el nivel t ($t = 0, ..., T$) lo notaremos como $J(t)$. Sea $J(t,j)$ ($j = 1, ..., |J(t)|$) un vértice del nivel t, éste queda definido por la tupla $\{(t,j), \vec{q}(t,j), \pi(t,j), W(\pi(t,j)), LB\_R(t,j), \triangle_Q(X(\pi(t,j)))\}$, donde:

- $\vec{q}(t,j) = (q_1(t,j), ..., q_{|I|}(t,j))$ representa el vector de demanda satisfecha asociada al vértice.
- $\pi(t,j) = (\pi_1(t,j), \pi_2(t,j), ..., \pi_t(t,j))$ es la secuencia parcial de t unidades de producto asociadas con el vértice $J(t,j)$.
- $W(\pi(t,j))$ es la sobrecarga parcial generada por la secuencia $\pi(t,j)$.
- $LB\_R(t,j)$ es una cota inferior de la sobrecarga generada por los productos no secuenciados, $d_i - q_i(t,j)$ ($i = 1, ..., |I|$).
- $\triangle_Q(X(\pi(t,j)))$ es la no-regularidad de la producción generada por la secuencia $\pi(t,j)$.

Obviamente, para obtener una cota global de la sobrecarga asociada al vértice $J(t,j)$, basta con hacer la siguiente operación: $LB\_W(t,j) = W(\pi(t,j)) + LB\_R(t,j)$.
Además, el vértice $J(t,j)$ presenta las siguientes propiedades:

$$\sum_{i=1}^{|I|} q_i(t,j) = t \tag{2.106}$$

$$\lfloor t \cdot \dot{d}_i \rfloor \leqslant q_i(t,j) \leqslant \lceil t \cdot \dot{d}_i \rceil \qquad \forall i = 1, ..., |I| \tag{2.107}$$

En el nivel 0 del grafo hay un sólo vértice $J(0)$. Inicialmente, podríamos pensar que en el nivel t, $J(t)$ contiene los vértices asociados a todas las subsecuencias que pueden construirse con t productos satisfaciendo las propiedades (2.106) y (2.107), no obstante, es fácil reducir el cardinal que a priori puede presentar $J(t)$ estableciendo las siguientes relaciones de pseudo-dominancia:

- Definición 1. PSD_1: dadas las secuencias $\pi(t,j_1)$ y $\pi(t,j_2)$ asociadas a los vértices $J(t,j_1)$ y $J(t,j_2)$, entonces $\pi(t,j_1)$ pseudo-domina a $\pi(t,j_2)$ si:

$$\begin{aligned}\pi(t,j_1) \prec \pi(t,j_2) \Leftrightarrow [\vec{q}(t,j_1) = \vec{q}(t,j_2)] \wedge [LB\_W(t,j_1) \leqslant LB\_W(t,j_2)] \wedge \\ \wedge [\triangle_Q(X(\pi(t,j_1))) \leqslant \triangle_Q(X(\pi(t,j_2)))]\end{aligned} \tag{2.108}$$

- Definición 2. PSD_2: dadas las secuencias $\pi(t,j_1)$ y $\pi(t,j_2)$ asociadas a los vértices $J(t,j_1)$ y $J(t,j_2)$, entonces $\pi(t,j_1)$ pseudo-domina a $\pi(t,j_2)$ si:

$$\begin{aligned}\pi(t,j_1) \prec \pi(t,j_2) \Leftrightarrow \{[\vec{q}(t,j_1) = \vec{q}(t,j_2)] \wedge [LB\_W(t,j_1) < LB\_W(t,j_2)]\} \vee \\ \vee \{[(\vec{q}(t,j_1), LB\_W(t,j_1)) \equiv (\vec{q}(t,j_2), LB\_W(t,j_2))] \wedge \\ \wedge [\triangle_Q(X(\pi(t,j_1))) \leqslant \triangle_Q(X(\pi(t,j_2)))]\}\end{aligned} \tag{2.109}$$

La reducción de $J(t)$ a través de las pseudo-dominancias definidas en (2.108) y (2.109) no puede garantizar la optimalidad de las soluciones.

Cotas para el problema

Dado un vértice de la etapa t alcanzado a través de la secuencia parcial $\pi(t,j) = \{\pi_1(t,j), ..., \pi_t(t,j)\}$, la cota global para W y la cota parcial del complemento $R(t,j)$ asociada con la secuencia o segmento $\pi(t,j)$ puede determinarse de acuerdo al esquema presentado en la Figura 2.12. Para obtener

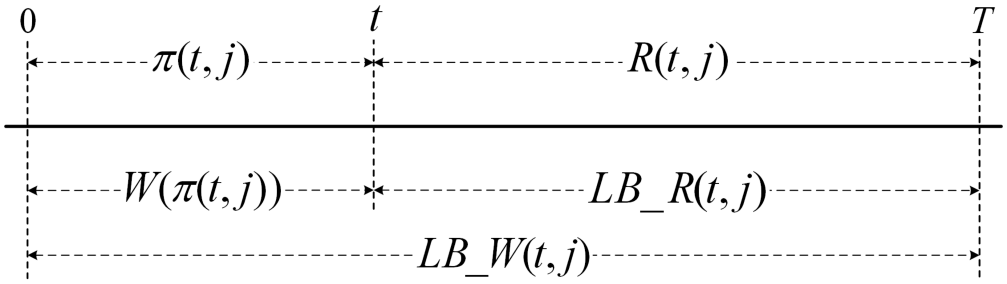

Figura 2.12: Esquema de acotación para la secuencia parcial $\pi(t,j)$ en el vértice $J(t,j)$

la sobrecarga asociada con $\pi(t,j)$, en cada etapa del procedimiento *BDP*, se usa un modelo matemático, al cual se van añadiendo las restricciones necesarias en cada una de las etapas. Dada una subsecuencia $\pi(t,j) = \{\pi_1(t,j), ..., \pi_t(t,j)\}$ de productos, los tiempos de proceso para cada estación $k \in K$ de la τ-*ésima* ($\tau = 1, ..., t$) unidad de la subsecuencia $\pi(t,j)$ son $\rho_{k,\tau} = p_{\pi_\tau(t,j),k}$ son conocidos. En estas condiciones, podemos formular el modelo matemático $LP\_W(\pi(t,j))$ en el cual se han suprimido las variables de asignación:

$$Min \quad W(\pi(t,j)) = \sum_{k=1}^{|K|} \left(b_k \sum_{\tau=1}^{t} w_{k,\tau} \right) \tag{2.110}$$

Sujeto a:

$$\rho_{k,\tau} = p_{\pi_\tau(t,j),k} \qquad k = 1, ..., |K|; \tau = 1, ..., t \tag{2.111}$$

$$\rho_{k,\tau} - w_{k,\tau} \geqslant 0 \qquad k = 1, ..., |K|; \tau = 1, ..., t \tag{2.112}$$

$$\hat{s}_{k,\tau} \geqslant \hat{s}_{k,\tau-1} + \rho_{k,\tau-1} - w_{k,\tau-1} - c \qquad k = 1, ..., |K|; \tau = 2, ..., t \tag{2.113}$$

$$\hat{s}_{k,\tau} \geqslant \hat{s}_{k-1,\tau} + \rho_{k-1,\tau} - w_{k-1,\tau} - c \qquad k = 2, ..., |K|; \tau = 1, ..., t \tag{2.114}$$

$$\hat{s}_{k,\tau} + \rho_{k,\tau} - w_{k,\tau} \leqslant l_k \qquad k = 1, ..., |K|; \tau = 1, ..., t \tag{2.115}$$

$$\hat{s}_{k,\tau} \geqslant 0; w_{k,\tau} \geqslant 0 \qquad k = 1, ..., |K|; \tau = 1, ..., t \tag{2.116}$$

$$\hat{s}_{1,1} = 0 \tag{2.117}$$

El resultado del modelo matemático presentado, $LP\_W(\pi(t,j))$, corresponde con $W(\pi(t,j))$.
Para obtener una cota de la sobrecarga asociada con el complemento $R(t,j)$ se usa una combinación de tres cotas inferiores:

- Dada una estación de trabajo k y el vértice $J(t,j)$, el tiempo disponible para completar las operaciones pendientes, para cada procesador homogéneo a actividad normal es:

$$TD_k(t,j) = (T-t-1)\cdot c + l_k \qquad k = 1,...,|K| \tag{2.118}$$

y el tiempo requerido para realizar dichas operaciones es:

$$TP_k(t,j) = \sum_{i=1}^{|I|} p_{i,k}\cdot(d_i - q_i(t,j)) \qquad k = 1,...,|K| \tag{2.119}$$

Usando (2.118) y (2.119), podemos definir una cota inferior de la sobrecarga total $R(t,j)$ como:

$$LB1(t,j) = \sum_{k=1}^{|K|} b_k \cdot [TP_k(t,j) - TD_k(t,j)]^+ \tag{2.120}$$

- Sin embargo, si se considera la mínima sobrecarga que un producto de tipo i puede generar, tenemos: :

$$LB2(i) = \left[\sum_{k=1}^{|K|} b_k(p_{i,k}-c) - b_{|K|}(l_{|K|}-c)\right]^+ \qquad i = 1,...,|I| \tag{2.121}$$

De esta manera, una cota para la sobrecarga del segmento $R(t,j)$ es:

$$LB2(t,j) = \sum_{i=1}^{|I|}(d_i - q_i(t,j))\cdot LB2(i) \tag{2.122}$$

- Una cota más refinada de la sobrecarga mínima que una unidad de producto de tipo i puede generar se puede obtener usando el siguiente modelo matemático:

$$LP\_LB3(i): \ Min \quad LB3(i) = \sum_{k=1}^{|K|} b_k \cdot w_{k,i} \tag{2.123}$$

Sujeto a:

$$\hat{s}_{k,i} \geqslant \hat{s}_{k-1,i} + p_{k-1,i} - w_{k-1,i} - c \qquad k = 2,...,|K| \tag{2.124}$$

$$\hat{s}_{k,i} + p_{k,i} - w_{k,i} \leqslant l_k \qquad k = 1,...,|K| \tag{2.125}$$

$$p_{k,i} - w_{k,i} \geqslant 0 \qquad k = 1,...,|K| \tag{2.126}$$

$$\hat{s}_{k,i} \geqslant 0; w_{k,i} \geqslant 0 \qquad k = 1,...,|K| \tag{2.127}$$

$$\hat{s}_{1,i} = 0 \tag{2.128}$$

Usando las soluciones del programa matemático previo, es posible determinar la siguiente cota para la sobrecarga de $R(t,j)$:

$$LB3(t,j) = \sum_{i=1}^{|I|}(d_i - q_i(t,j))\cdot LB3(i) \tag{2.129}$$

Finalmente es posible determinar la cota $LB\_R(t,j)$ para el complemento de la siguiente manera:

$$LB\_R(t,j) = max\left\{LB1(t,j), LB2(t,j), LB3(t,j)\right\} \tag{2.130}$$

Y finalmente, una cota inferior para la sobrecarga total asociada al vértice $J(t,j)$ se puede obtener con la siguiente relación:

$$LB\_W(t,j) = W(\pi(t,j)) + LB\_R(t,j) \tag{2.131}$$

Propiedades derivadas de las restricciones en el mix de producción (*pmr*)

En este apartado, se estudian las propiedades de las secuencias de productos que derivan de la incorporación de las restricciones para preservar el mix de producción (*pmr*) en el *MMSP-W*.
Primero de todo, es necesario definir como se mide la no-regularidad de la producción ($\triangle_Q(X)$) en cada vértice del grafo asociado con el problema. Dado un vértice $J(t,j)$, asociado con una secuencia $\pi(t,j) = \{\pi_1(t,j), ..., \pi_t(t,j)\}$, sea $X_{i,\tau}(\pi(t,j))$ $(i = 1, ..., |I|, \tau = 1, ..., t)$ el número de unidades del tipo de producto i secuenciado en las primeras τ unidades de la secuencia $\pi(t,j)$:

$$X_{i,\tau}(\pi(t,j)) = |\left\{\pi_h(t,j) \in \pi(t,j) : (\pi_h(t,j) = \{i\}) \wedge (1 \leqslant h \leqslant \tau)\right\}| \tag{2.132}$$

Usando la defición (2.132) podemos definir la no-regularidad de la producción de la secuencia asociada a $\pi(t,j)$ del vértice $J(t,j)$ como:

$$\triangle_Q(X(\pi(t,j))) = \sum_{\tau=1}^{t}\sum_{i=1}^{|I|} X_{i,\tau}(\pi(t,j)) - \tau \cdot \dot{d}_i)^2 \tag{2.133}$$

Las restricciones para preservar el mix de producción se pueden expresar como sigue:

$$\lfloor t \cdot \dot{d}_i \rfloor \leqslant X_{i,t} \leqslant \lceil t \cdot \dot{d}_i \rceil \qquad\qquad i = 1, ..., |I|; t = 1, ..., T \tag{2.134}$$

donde $X_{i,t}$ es una variable que representa el número total de unidades del tipo de producto i secuenciadas durante los t primeros ciclos de producción.
Si imponemos estas restricciones a las secuencias, podemos derivar las siguientes propiedades:

Teorema 2. *Si* $\lfloor t \cdot \dot{d}_i \rfloor \leqslant X_{i,t} \leqslant \lceil t \cdot \dot{d}_i \rceil, (\forall i \in I; \forall t)$, *entonces* $X_{i,t} - X_{j,t} \leqslant \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_j \rfloor, (\forall \{i,j\} \subseteq I; \forall t)$.

Demostración. Obviamente, se debe satisfacer: $\left(X_{i,t} \leqslant \lceil t \cdot \dot{d}_i \rceil\right) \wedge \left(\lfloor t \cdot \dot{d}_j \rfloor \leqslant X_{j,t}\right)$ $(\forall \{i,j\} \subseteq I; \forall t)$. Entonces $X_{i,t} + \lfloor t \cdot \dot{d}_j \rfloor \leqslant \lceil t \cdot \dot{d}_i \rceil + X_{j,t} \Leftrightarrow X_{i,t} - X_{j,t} \leqslant \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_j \rfloor, (\forall \{i,j\} \subseteq I; \forall t)$. □

Corolario 1. *Si* $d_i \leqslant d_j$, *entonces* $X_{i,t} - X_{j,t} \leqslant 1, (\forall \{i,j\} \subseteq I; \forall t)$.

Demostración. Del Teorema 2, tenemos $X_{i,t} - X_{j,t} \leqslant \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_j \rfloor$. En cambio, $d_i \leqslant d_j \Rightarrow \lfloor t \cdot \dot{d}_i \rfloor \leqslant \lfloor t \cdot \dot{d}_j \rfloor$. Finalmente, podemos escribir: $X_{i,t} - X_{j,t} \leqslant \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_j \rfloor \leqslant \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_i \rfloor \leqslant 1, (\forall \{i,j\} \subseteq I; \forall t)$. □

Teorema 3. *Si* $\lfloor t\cdot\dot{d}_i\rfloor \leqslant X_{i,t} \leqslant \lceil t\cdot\dot{d}_i\rceil, (\forall i;\forall t)$, *entonces* $X_{i,t}-X_{j,t}\geqslant\lfloor t\cdot\dot{d}_i\rfloor-\lceil t\cdot\dot{d}_j\rceil, (\forall\{i,j\}\subseteq I;\forall t)$.

Demostración. Se debe satisfacer: $\left(X_{i,t}\geqslant\lfloor t\cdot\dot{d}_i\rfloor\right)\wedge\left(\lceil t\cdot\dot{d}_j\rceil\geqslant X_{j,t}\right), (\forall\{i,j\}\subseteq I;\forall t)$. Entonces, $X_{i,t}+\lceil t\cdot\dot{d}_j\rceil\geqslant\lfloor t\cdot\dot{d}_i\rfloor+X_{j,t}\iff X_{i,t}-X_{j,t}\geqslant\lfloor t\cdot\dot{d}_i\rfloor-\lceil t\cdot\dot{d}_j\rceil\ (\forall\{i,j\}\subseteq I;\forall t)$. □

Corolario 2. *Si* $d_i\geqslant d_j$ *entonces* $X_{i,t}-X_{j,t}\leqslant -1, (\forall\{i,j\}\subseteq I;\forall t)$.

Demostración. Del Teorema 3, tenemos $X_{i,t}-X_{j,t}\geqslant\lfloor t\cdot\dot{d}_i\rfloor-\lceil t\cdot\dot{d}_j\rceil$. Además, $d_i\geqslant d_j\Rightarrow\lceil t\cdot\dot{d}_i\rceil\geqslant\lceil t\cdot\dot{d}_j\rceil$. Finalmente, podemos escribir: $X_{i,t}-X_{j,t}\geqslant\lfloor t\cdot\dot{d}_i\rfloor-\lceil t\cdot\dot{d}_j\rceil\geqslant\lfloor t\cdot\dot{d}_i\rfloor-\lceil t\cdot\dot{d}_i\rceil\geqslant -1$, $(\forall\{i,j\}\subseteq I;\forall t)$. □

Corolario 3. *Si* $d_i=d_j$ *entonces* $|X_{i,t}-X_{j,t}|\leqslant 1, (\forall\{i,j\}\subseteq I;\forall t)$.

Demostración. Obviamente, del Corolario 1, $X_{i,t}-X_{j,t}\leqslant 1, (\forall\{i,j\}\subseteq I;\forall t)$. Del Corolario 2, $X_{i,t}-X_{j,t}\leqslant -1, (\forall\{i,j\}\subseteq I;\forall t)$. Entonces, $|X_{i,t}-X_{j,t}|\leqslant 1, (\forall\{i,j\}\subseteq I;\forall t)$. □

El cumplimiento de las restricciones *pmr* combinados con la variedad de la demanda resulta en la siguiente propiedad:

Teorema 4. *Si* $\lfloor t\cdot\dot{d}_i\rfloor\leqslant X_{i,t}\leqslant\lceil t\cdot\dot{d}_i\rceil, (\forall i;\forall t)$, *dada la secuencia* $\pi=\{\pi_1,\pi_2,...,\pi_T\}$, *donde* $\pi_t=\{j\}$ *con* $2\leqslant t\leqslant T$, *se satisface: Si* $\exists i\in I:(X_{i,t}>0)\wedge(d_i\leqslant d_j)\Rightarrow X_{i,t}\leqslant X_{j,t}, (\forall t=2,...,T)$.

Demostración. Si suponemos que $\exists i\in I:(X_{i,t}>0)\wedge(d_i\leqslant d_j)$ tal que $X_{i,t}>X_{j,t}$, entonces $X_{i,t}-X_{j,t}\geqslant 1$.
En cambio, dado $\pi_t=\{j\}$, se debe satisfacer: $X_{j,t}=X_{j,t-1}+1$ y $X_{i,t}=X_{i,t-1}$. Entonces podemos escribir $X_{i,t}-X_{j,t}=X_{i,t-1}-X_{j,t-1}-1\geqslant 1\Rightarrow X_{i,t-1}-X_{j,t-1}\geqslant 2$.
Además, dado que $X_{i,t-1}-X_{j,t-1}\leqslant\lceil(t-1)\cdot\dot{d}_i\rceil-\lfloor(t-1)\cdot\dot{d}_j\rfloor\leqslant\lceil(t-1)\cdot\dot{d}_j\rceil-\lfloor(t-1)\cdot\dot{d}_j\rfloor$, tenemos $\lceil(t-1)\cdot\dot{d}_j\rceil-\lfloor(t-1)\cdot\dot{d}_j\rfloor\geqslant X_{i,t-1}-X_{j,t-1}\geqslant 2$, lo cual es absurdo. Entonces, la hipótesis $X_{i,t}>X_{j,t}$ es falsa y consecuentemente, se debe satisfacer $X_{i,t}\leqslant X_{j,t}, (\forall t=2,...,T)$ y $\forall i\in I:X_{i,t}>0$, cuando $\pi_t=\{j\}$. □

Corolario 4. *Si* $\lfloor t\cdot\dot{d}_i\rfloor\leqslant X_{i,t}\leqslant\lceil t\cdot\dot{d}_i\rceil, (\forall i\subseteq I;\forall t)$ *dada la secuencia* $\pi=\{\pi_1,\pi_2,...,\pi_T\}$, *donde* $\pi_t=\{j\}$ *con* $2\leqslant t\leqslant T$, *y si* $\exists i\in I:X_{i,t}>X_{j,t}\Rightarrow d_i>d_j$.

Demostración. Evidentemente, del Teorema 4, $d_i\leqslant d_j\Rightarrow X_{i,t}\leqslant X_{j,t}$, lo cual niega la hipótesis $(X_{i,t}>X_{j,t})$; por lo tanto, se debe cumplir que $d_i>d_j$. □

Reglas para descartar vértices

En la etapa t, sea $X_i(\pi(t-1,h))$ la demanda satisfecha para el tipo de producto $i\in I$ asociado con la secuencia $\pi(t-1,h)$ del vértice $J(t-1,h)$. Asumamos que una extensión del vértice $J(t-1,h)$ se construye añadiendo en la etapa t un producto tipo j de la secuencia, sea $J(t,h')$ el vértice

resultante para la secuencia parcial $\pi(t,h') = \pi(t-1,h) \cup \{j\}$. Las demandas satisfechas deben cumplir:

$$\begin{aligned} X_i(\pi(t,h')) &= X_i(\pi(t-1,h)) && \forall i \neq j \\ X_j(\pi(t,h')) &= X_j(\pi(t-1,h)) + 1 && con\ \pi_t(t,h') = \{j\} \end{aligned}$$

Bajo estas condiciones, el vértice $J(t,h')$ puede ser descartado del proceso de exploración si cualquiera de las siguientes reglas se cumple:

- *Bloque I*: Restricciones *pmr*.
  $\forall j \in I,\ Si\ \exists j : \left[X_j(\pi(t,h')) < \lfloor t \cdot \dot{d}_j \rfloor\right] \vee \left[X_j(\pi(t,h')) > \lceil t \cdot \dot{d}_j \rceil\right] \rightarrow Descartar\ J(t,h')$

- *Bloque II*: Teoremas 2 y 3.
  $\forall i \neq j \in I,\ Si\ \exists i : \left[X_i(\pi(t,h')) - X_j(\pi(t,h')) > \lceil t \cdot \dot{d}_i \rceil - \lfloor t \cdot \dot{d}_j \rfloor\right] \vee$
  $\vee \left[X_i(\pi(t,h')) - X_j(\pi(t,h')) < \lfloor t \cdot \dot{d}_i \rfloor - \lceil t \cdot \dot{d}_j \rceil\right] \rightarrow Descartar\ J(t,h')$

- *Bloque III*: Corolarios 1, 2 y 3.
  $\forall i \neq j \in I,\ Si\ [[d_i < d_j] \wedge [X_i(\pi(t,h')) - X_j(\pi(t,h')) > 1]] \vee$
  $\vee Si\ [[d_i = d_j] \wedge [|X_i(\pi(t,h')) - X_j(\pi(t,h'))| > 1]] \vee$
  $\vee Si\ [[d_i > d_j] \wedge [X_j(\pi(t,h')) - X_i(\pi(t,h')) > 1]] \rightarrow Descartar\ J(t,h')$

- *Bloque IV*: Teorema 4.
  Dada la secuencia parcial $\pi(t,h')$ asociada al vértice $J(t,h')$, con $\pi_t(t,h') = \{j\}$:
  $\forall i \in I : X_i(\pi(t,h')) \geqslant 1;\ Si\ [[d_j \geqslant d_i] \wedge [X_j(\pi(t,h')) < X_i(\pi(t,h'))]] \rightarrow Descartar\ J(t,h')$

El uso de la *BDP*

El procedimiento escogido basado en *BDP* presenta el siguiente esquema algorítmico, al que se le han incorporado las nuevas cotas propuestas (a partir de la empleo de un modelo matemático que se amplia etapa a etapa) y las reglas para descartar vértices descritas en el apartado anterior (para más detalles del procedimiento *BDP* ver apartado 2.3.6):

Algoritmo 5 *Esquema BDP para el problema MMSP-W con instantes de interrupción libres, restricciones* pmr *y reglas de descarte de vértices*

Entrada: $T, |I|, |K|, d_i(i = 1, ..., |I|), p_{i,k}(i = 1, ..., |I|; k = 1, ..., |K|), l_k, b_k(\forall k), c, Z_0, H$
Salida: Lista de secuencias obtenidas con *BDP*

1: Inicialización: $t = 0; LBZ_{min} = \infty$
2: Generar_modelo()
3: **mientras** $t < T$ **hacer**
4: $t \leftarrow t + 1$
5: Añadir_restricciones(t)
6: **mientras** lista de vértices consolidados en la etapa $t - 1$ **no vacía hacer**
7: Seleccionar_vértice(t)
8: Desarrollar_vértice(t)
9: Filtrar_vértices(Z_0, H, LBZ_{min})
10: **fin mientras**
11: Finalizar_etapa()
12: **fin mientras**

En el procedimiento aparecen las siguientes funciones:

- Generar_modelo(): Esta función genera el modelo inicial $LP\_W(\pi(t, j))$ para obtener la solución óptima para $W^*(\pi(t, j))$ para $t = 0$.
- Añadir_restricciones(t): Esta función añade las nuevas restricciones al modelo $LP\_W(\pi(t, j))$ asociadas a la nueva etapa t.
- Seleccionar_vértice(t): selecciona, siguiendo un orden no decreciente de los valores $LB\_W(t, j)$, uno de los vértices consolidados en la etapa $t - 1$.
- Desarrollar_vértice(t): desarrolla el vértice seleccionado en el paso anterior añadiendo un nuevo producto con demanda pendiente. Los vértices que no satisfacen las propiedades (2.106) y (2.107) no se generan. Esto se realiza incorporando las reglas descritas en el apartado anterior (Bloques de reglas I, II, III y IV). Como último paso, se calcula la cota $LB\_W(t, j)$ usando el modelo matemático $LP\_W(\pi(t, j))$ para calcular $W^*(\pi(t, j))$.
- Filtrar_vértices(Z_0, H, LBZ_{min}): escoge, de todos los vértices desarrollados en el paso anterior, un número máximo H de los vértices más prometedores (de acuerdo con el menor valor de la cota inferior $LB\_W(t, j)$). Además, se eliminan los vértices cuya cota inferior sea igual o superior a una solución conocida (Z_0), se rechazan los vértices generados pseudo-dominados por aquellos vértices ya escogidos previamente y se desplazan los vértices escogidos previamente pseudo-dominados por los nuevos vértices, como se define en (2.108) y (2.109).

- Finalizar_etapa(): consolida los vértices más prometedores de la etapa t (como máximo un número H de éstos).

Un ejemplo de la reducción del grafo

La Figura 2.13 representa la exploración de vértices del grafo asociado al problema para resolver el ejemplo propuesto en el apartado 2.5.2 usando el procedimiento *BDP* detallado anteriormente. Sin embargo, es esta figura no se realiza ninguna de las eliminaciones de vértices permitidas por la incorporación de las restricciones *pmr* al *MMSP-W*. Para este ejemplo, se ha utilizado como solución inicial $Z_0 = 4$ y un ancho de ventana $H = 6$.
La Figura 2.14 presenta la misma exploración cuando se emplean las reglas de eliminación de vértices dispuestas en los Bloques I, II, III y IV del apartado 2.5.3 para asegurar el cumplimiento de las restricciones *pmr* en el procedimiento *BDP*. De nuevo, para este grafo se ha utilizado $Z_0 = 4$ y $H = 6$ (aunque fue suficiente con $H = 3$). En las Figuras 2.13 y 2.14 se pueden observar los siguientes estados de eliminación de vértices:

1. Vértice dominado (*d*). Por ejemplo, en la Figura 2.13 , el vértice representante de la secuencia parcial (B, C), con $\triangle_Q(X) = 2{,}6$, está dominado por (C, B), con $\triangle_Q(X) = 2{,}3$. Además, en la misma Figura, se puede observar que (B, A) está dominado por (A, B).

2. Vértice eliminado (*r*). La limitación del ancho de ventana a $H = 6$ contribuye a seleccionar los vértices más prometedores (mejor valor para $LB\_W(t, j)$ para ser desarrollado en cada etapa t. Por ejemplo, en la etapa $t = 3$ de la Figura 2.13, los vértices que corresponden con las secuencias parciales (A, A, A), (A, C, C), (C, B, C) y (C, C, B) son eliminados y sólo seis vértices son desarrollados para alcanzar la etapa $t = 4$.

3. Vértice desestimado (Z_0). Los vértices desestimados son aquellos cuyo desarrollo no puede finalizar en una solución mejor que la mejor solución conocida Z_0. Por ejemplo, la secuencia (A, C, B, A, A, C) en la Figura 2.14 no mejora la mejor solución conocida $Z_0 = 4$.

4. Vértice ilegal (*pmr*). Este es un vértice para el cual la secuencia no satisface las restricciones *pmr*. Por ejemplo, en la Figura 2.14, las secuencias parciales (A, A), (B, C), (C, B) y (C, C) no satisfacen las restricciones *pmr*.

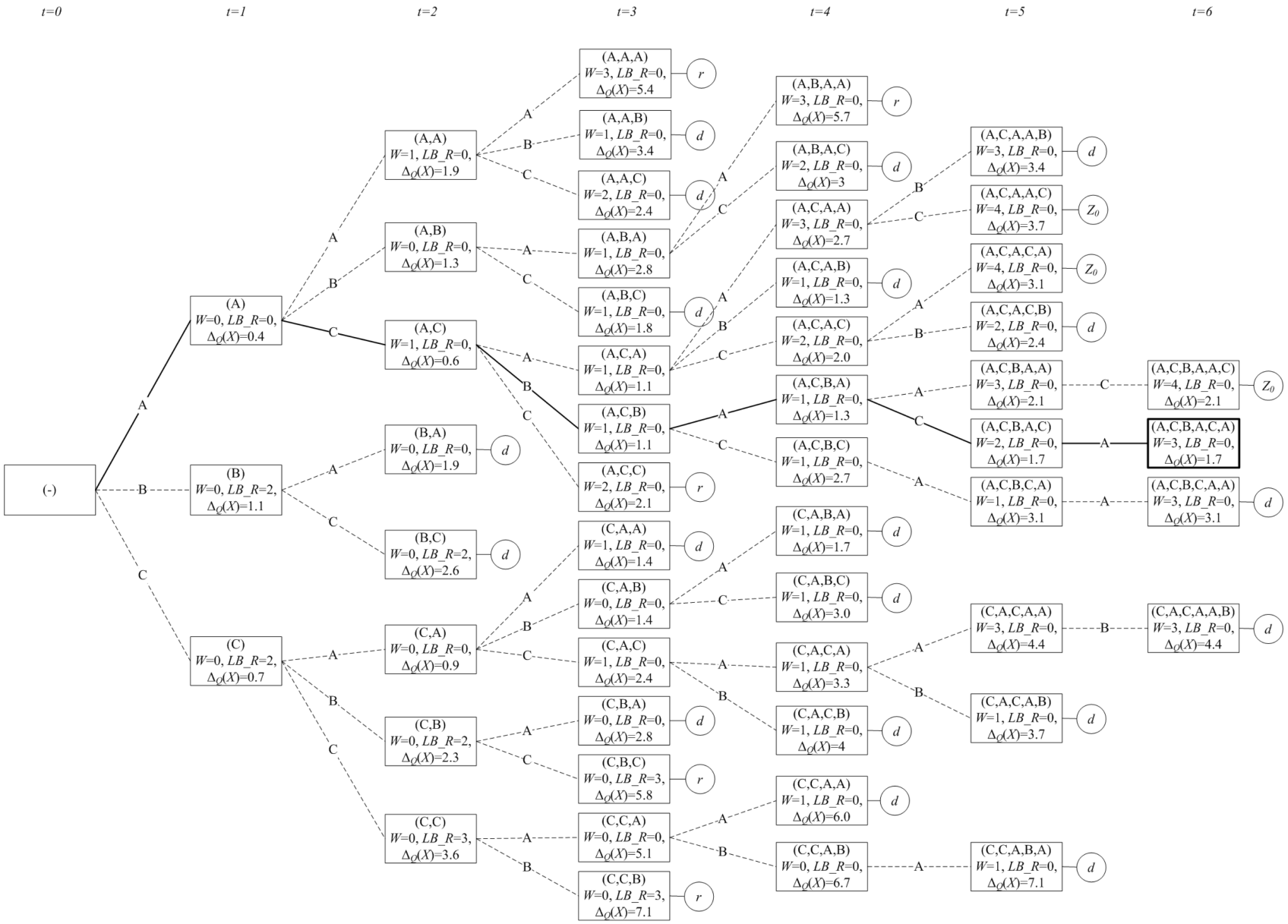

Figura 2.13: Grafo original para el ejemplo usando $Z_0 = 4$, pseudo-dominancias 1 (PSD_1) y $H = 6$

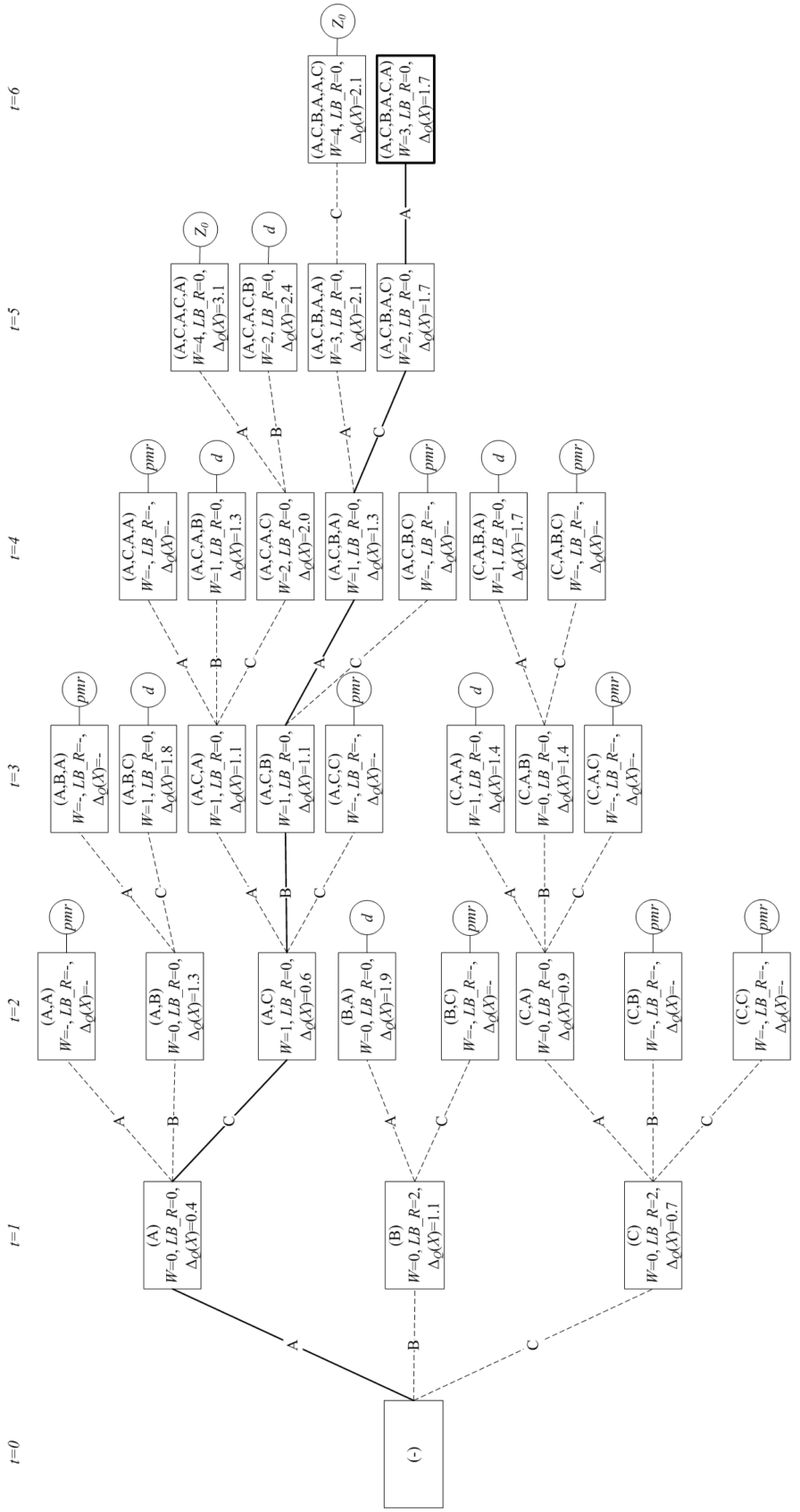

Figura 2.14: Grafo usando las restricciones *pmr*, $Z_0 = 4$, pseudo-dominancias 1 (PSD_1) y $H = 6$

2.5.4. Experiencia computacional

Experiencia computacional con instancias de referencia

Para la comprobación de la calidad del nuevo procedimiento *BDP*, se han utilizado las 225 instancias presentadas en el apartado 2.3.7.
Las soluciones obtenidas por el procedimiento *BDP* propuesto en este apartado, al que denominaremos *BDP-2*, se han obtenido usando las siguientes condiciones:

1. El procedimiento *BDP-2* está programado en C++, usando gcc 4.2.1 como compilador. El código se ha ejecutado en un Apple iMac con un procesador Intel Core i7 a 2.93 GHz y 8 Gb de RAM, usando MAC OS X 10.6.7 (sin ningún tipo de código en paralelo, por tanto el ordenador funcionó como un único procesador a 2.93 GHz).

2. Se han usado 4 anchos de ventana (H) cuyos valores fueron 1, 6, 16 y 32.

3. La solución inicial Z_0 para cada ancho de ventana fue la solución obtenida con *BDP-2* en el ancho de ventana previo, excepto en el caso $H = 1$ donde $Z_0 = \infty$.

4. Para calcular las cotas inferiores, $LB\_W(t, j)$, de la sobrecarga asociada con cada vértice en el procedimiento *BDP-2*, se ha usado el solver Gurobi (v4.6.1) para resolver el programa lineal $LP\_W(\pi(t, j))$.

Para contrastar la calidad del procedimiento *BDP-2* se han usado las siguientes herramientas:

1. Gurobi: Se resolvió el modelo $M4\_pmr$ usando el solver Gurobi (resultados disponibles en (Bautista, Cano and Alfaro, 2012c)).

2. *BDP-1*: este es un procedimiento basado en *BDP*, muy similar al presentado en el apartado 2.4, al que se le han incorporado las restricciones *pmr* para el cálculo de las cotas (detalles disponibles en (Bautista, Cano and Alfaro, 2012b)).

3. *BDP-2*: El procedimiento *BDP* descrito en este apartado. Para reducir el espacio de búsqueda se han implementado los dos mecanismos basados en las pseudo-dominancias (ver (2.108) y (2.109)) entre vértices. Las pseudo-dominancias *PSD_1*(2.108) implican que un vértice es descartado en una etapa cuando otro vértice (con idéntica demanda satisfecha) existe en la misma etapa y tiene un mejor valor para sobrecarga parcial y no-regularidad de la producción. En cambio, las pseudo-dominancias *PSD_2*(2.109) implican que un vértice pseudo-dominada a otro sólo si presenta un mejor valor de sobrecarga parcial, y si este es igual, descartaremos aquel que presente peor valor para la no-regularidad del mix de producción.

Para comprobar el comportamiento de los tres procedimientos, teniendo en cuenta el conjunto de instancias E y designando como $\hat{S}_G(\epsilon)$ (óptima para W), $\hat{S}_1(\epsilon)$ y $\hat{S}_2(\epsilon)$ las mejores soluciones

obtenidas para las instancias $\epsilon \in E$ utilizando los procedimientos Gurobi, *BDP-1* y *BDP-2* (que presenta dos variantes, *BDP-2/1* y *BDP-2/2*, según si utilizan las pseudo-dominacias *PSD_1*(2.108) o *PSD_2*(2.109), respectivamente), se han usado las siguientes definiciones de desviación porcentual relativa:

$$RPD_1(f,\epsilon) = \frac{f(\hat{S}_G(\epsilon)) - f(\hat{S}_2(\epsilon))}{f(\hat{S}_G(\epsilon))} \qquad f \in \{W, \triangle_Q(X)\}; \epsilon \in E \tag{2.135}$$

$$RPD_2(f,\epsilon) = \frac{f(\hat{S}_1(\epsilon)) - f(\hat{S}_2(\epsilon))}{f(\hat{S}_1(\epsilon))} \qquad f \in \{W, \triangle_Q(X)\}; \epsilon \in E \tag{2.136}$$

Los resultados del experimento están resumidos en las Tablas 2.17 y 2.18:

Tabla 2.17: Mínimo, Máximo y promedio de tiempos de CPU necesarios para obtener las soluciones para las 225 instancias usando Gurobi, *BDP-1* y *BDP-2* (ambas variantes)

	Gurobi	*BDP-1*	*BDP-2/1*	*BDP-2/2*
min CPU	0.03	0.06	0.02	0.02
max CPU	110.53	2.72	0.17	0.07
$\overline{CPU}$	11.79	0.78	0.06	0.05

Tabla 2.18: Valores RPD_1 y RPD_2 para las estructuras y bloques y promedio (de las 225 instancias) para las soluciones obtenidas usando Gurobi, *BDP-1* y *BDP-2* (ambas variantes)

	W				$\triangle_Q(X)$			
	BDP-2/1		*BDP-2/2*		*BDP-2/1*		*BDP-2/2*	
	RPD_1	RPD_2	RPD_1	RPD_2	RPD_1	RPD_2	RPD_1	RPD_2
E1	-3.15	-3.05	-5.73	-5.63	2.14	7.01	-15.37	-10.01
E2	-1.86	-1.84	-5.18	-5.16	10.84	9.6	-23.58	-25.28
E3	-0.49	-0.49	-2.28	-2.28	13.07	12.23	-10.59	-11.81
E4	-0.08	-0.08	-0.32	-0.32	4.23	3.85	0.32	-0.5
E5	-1.65	-1.64	-9.25	-9.23	2.89	2.99	-32.19	-33.04
B1	-0.86	-0.86	-1.55	-1.55	-1.23	0.84	-16.16	-14.11
B2	-0.58	-0.58	-2.2	-2.2	1.8	3.77	-10.06	-8.08
B3	-1.62	-1.62	-6.11	-6.11	6.09	6.5	-8.8	-8.32
B4	-1.93	-1.81	-3.02	-2.9	1.71	3.14	-10.18	-8.78
B5	-1.63	-1.6	-5.44	-5.41	10.13	9.87	-21.06	-21.98
Promedio	-1.45	-1.42	-4.55	-4.52	6.63	7.13	-16.28	-16.13

Como se muestra en la Tabla 2.17, se ha obtenido una substancial mejora en el tiempo de CPU promedio usando el procedimiento *BDP-2* respecto a los otros dos. Especificamente, *BDP-1* re-

quiere 13 veces y 15 veces más tiempo que *BDP-2/1* y *BDP-2/2*, respectivamente, para alcanzar las mejores soluciones de las 225 instancias. Adicionalmente, Gurobi requirió 196 y 235 veces más tiempo en promedio que *BDP-2/1* y *BDP-2/2*. En términos de máxima CPU, *BDP-2/1* y *BDP-2/2* alcanzaron mejoras de 650 y 1579 veces al tiempo necesario por Gurobi. Cuando se compara ambos procedientos *BDP-2* con *BDP-1*, estos alcanzaron una mejora de 16 y 38 veces, respectivamente. En lo que respecta a los tiempos mínimos de CPU, no existe diferencia apreciable entre los tres procedimientos.
En la Tabla 2.18 se puede observar una mejora en la regularidad de la producción obtenida usando los procedimientos *BDP-2*. Específicamente, *BDP-2/1* reduce, en promedio, la función de no-regularidad $\triangle_Q(X)$ en un 6.63 % comparada con las obtenidas por Gurobi y un 7.13 % en comparación con *BDP-1*. Sin embargo, en promedio, la sobrecarga de trabajo W de *BDP-2/1* es 1.45 % y 1.42 % peor que la de Gurobi y *BDP-1*, respectivamente. Estas mejoras en regularidad de la producción y incremento en la sobrecarga de trabajo también ocurren monótonamente cuando agrupamos las instancias en bloques y estructuras. La mejora más significativa en regularidad se encuentra en la estructura *E3* y en el bloque *B5*, mientras que el incremento en sobrecarga de trabajo, aunque más balanceado entre bloques y estructuras, son más relevantes en *E1* y *B4*. En contraste, el procedimiento *BDP-2/2*, que incorpora las pseudo-dominancias *PSD_2*, no aporta mejoras cuando se compara con procedimientos previos.

Instancias de estudio en la Planta de motores Nissan-BCN

Dada la mejoría en tiempos de CPU obtenidas con los nuevos procedimientos *BDP-2*, se ha realizado una experiencia computacional con las instancias pertenecientes al caso de estudio de la planta de motores de Nissan-BCN (ver apartado 2.3.7) para comprobar la calidad de los procedimientos *BDP-2/1* y *BDP-2/2*.
En las Tablas 2.7 y 2.8 se encuentran los detalles de los tiempos de proceso de los nueve motores en las 21 estaciones de trabajo, así como los diferentes planes de demanda. De nuevo, se ha utilizado un tiempo de ciclo $c = 175s$. La ventana temporal escogida también ha sido $l_k = 195s,\ \forall k$, y idéntica para todas las estaciones, ya que esta ventana temporal nos asegura un margen de seguridad sobre el tiempo de ciclo de un 10 %. Con estos datos, las instancias con una demanda total $T = 270$ están asociadas con un único día de trabajo con un tiempo efectivo de trabajo de $13{,}125$ horas distribuidas en dos turnos. Se ha considerado un número idéntico de procesadores homogéneos en cada estación $b_k = 1$; el procesador en cada estación está formado por dos trabajadores con habilidades idénticas y las herramientas y el equipamiento auxiliar necesario.
Para estudiar el comportamiento de los procedimientos *BDP-2*, se ha asumido diferentes planes de demanda (ver Tabla 2.8) asociados a un día de trabajo (los correspondientes al *Bloque I* de la Tabla). Los anchos de ventana escogidos en esta ocasión han sido $H = 1, 36, 81$ y 126, usando como mejor solución conocida Z_0 la solución obtenida en el ancho de ventana anterior, excepto en

$H = 1$ en el que se usó ∞. Los resultados obtenidos con *BDP-2/1* y *BDP-2/2* se han comparado con los obtenidos por Gurobi y se encuentran en las Tablas 2.19 y 2.20.

Tabla 2.19: Tiempos de CPU para el caso de estudio de la planta de motores de Nissan usando Gurobi y los procedimientos *BDP-2*

	Gurobi	*BDP-2/1*				*BDP-2/2*			
		H=1	*H=36*	*H=81*	*H=126*	*H=1*	*H=36*	*H=81*	*H=126*
min CPU	149.4	0.1	427.6	717.9	883.0	0.1	369.0	524.1	535.0
max CPU	7200.0	35.0	526.2	1119.9	1701.1	35.9	509.8	972.2	1134.2
$\overline{CPU}$	6605.1	3.2	484.5	992.9	1465.5	3.2	450.1	772.4	854.8

En lo que respecta a tiempos de CPU (ver Tabla 2.19), *BDP-2/1* y *BDP-2/2* usando un ancho de ventana de $H = 126$ (el más grande usado en este experimento) mejoraron los tiempos de CPU necesarios comparados con los que necesitó Gurobi en 4 y 7 veces, en promedio.
La Tabla 2.20 presenta los mejores valores para W y $\triangle_Q(X)$ de las 23 instancias alcanzados con Gurobi y los procedimientos *BDP-2/1* y *BDP-2/2*, para cuatro anchos de ventana ($H = 1, 36, 81, 126$). Los resultados más destacados son los siguientes:

1. Ambas versiones de *BDP-2* mejoraron, en promedio, las mejores soluciones para W y $\triangle_Q(X)$ obtenidas con Gurobi. Los valores para las mejoras obtenidas usando *BDP-2/1* para W y $\triangle_Q(X)$ fueron 6.78 % y 15.07 %, respectivamente. Usando *BDP-2/2*, las mejoras fueron solamente de 2.20 % y 0.37 %, para W y $\triangle_Q(X)$ respectivamente.

2. Comparando con Gurobi, *BDP-2/1* mejoró el valor de $\triangle_Q(X)$ para las 23 instancias y la sobrecarga de trabajo (W) en 16 instancias.

3. *BDP-2/1* dominó a *BDP-2/2* con respecto al valor $\triangle_Q(X)$. *BDP-2/2* obtuvo mejores valores para W que *BDP-2/1* en sólo 4 de las 23 instancias.

A la vista de los resultados de esta experiencia computacional y de la experiencia computacional previa, podemos concluir que *BDP-2/1* es más competitivo, en promedio, que el resto de procedimientos.

Tabla 2.20: Resultados para W y $\triangle_Q(X)$ del caso de estudio de la planta de motores de Nissan usando el sover Gurobi (limitado a 7200s) y los procedimientos *BDP-2*. Valores para RPD_1 para W y $\triangle_Q(X)$. El símbolo '*' denota una solución óptima

	Gurobi		*BDP-2/1*										*BDP-2/2*									
			H=1		*H=36*		*H=81*		*H=126*		*Best*		*H=1*		*H=36*		*H=81*		*H=126*		*Best*	
	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	RPD_1W	$RPD_1 \triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	W	$\triangle_Q(X)$	RPD_1W	$RPD_1 \triangle_Q(X)$
1	186	400	368	400	166	400	-	-	-	-	10.8	0	368	400	166	400	-	-	-	-	10.8	0
2	383	423.5	-	-	404	393.5	358	369.5	318	327.9	17	22.6	-	-	363	425.4	342	431.4	-	-	10.7	-0.4
3	423	408.5	-	-	444	340.7	-	-	-	-	-5	16.6	-	-	459	378	458	398.5	-	-	-8.3	7.5
4	307	421.3	-	-	352	392.6	305	333.6	-	-	0.7	20.8	-	-	352	395.5	-	-	-	-	-14.7	6.1
5	661	394.7	-	-	641	405.6	633	352.1	-	-	4.2	10.8	-	-	641	402.6	-	-	-	-	3	-2
6	478	420	-	-	467	385	451	344	447	324.3	6.5	22.8	-	-	482	388.5	428	394	-	-	10.5	7.5
7	731	396	-	-	741	388.1	-	-	-	-	-1.4	2	-	-	762	423.6	752	445	740	436.5	-1.2	-7
8	160	448.1	-	-	158	392.9	121	364.2	112	347.6	30	22.4	-	-	158	412	130	399.6	126	397.1	21.3	11.4
9	751	411.2	-	-	773	440.7	739	360.7	-	-	1.6	12.3	-	-	770	438.6	-	-	-	-	-2.5	-6.7
10	1208*	381.1	-	-	1209	330.8	-	-	-	-	-0.1	13.2	-	-	1209	410.9	-	-	-	-	-0.1	-7.8
11	122	447.3	-	-	99	403.6	96	384.1	92	384.4	24.6	14.1	-	-	96	430	-	-	-	-	21.3	3.9
12	287	410.2	-	-	334	425.1	293	385.5	-	-	-2.1	6	-	-	328	428.9	314	438.3	-	-	-9.4	-4.6
13	336	436.4	-	-	316	388.1	286	386.7	277	334.5	17.6	23.4	-	-	297	443.6	295	458.9	294	463.7	12.5	-1.6
14	423	414.9	-	-	426	411.3	383	353.9	381	354.9	9.9	14.7	-	-	417	445.7	409	427.1	407	446.1	3.8	-2.9
15	442	445.2	-	-	445	413.2	422	378.1	-	-	4.5	15.1	-	-	448	443.2	-	-	-	-	-1.4	0.4
16	251	404.9	-	-	245	394.9	232	353.9	216	340	13.9	16	-	-	240	413.4	217	418.3	-	-	13.5	-2.1
17	488	415.3	-	-	513	376.8	493	370.2	466	386.5	4.5	10.9	-	-	497	413.7	470	431.9	-	-	3.7	0.4
18	619	419.6	-	-	648	424.7	620	381.1	610	336.3	1.5	19.9	-	-	626	414.1	610	428.3	-	-	1.5	1.3
19	945*	412.3	-	-	950	336.7	-	-	-	-	-0.5	18.3	-	-	949	412.2	-	-	-	-	-0.4	0
20	150	393.6	-	-	153	390	138	342.6	129	344.6	14	13	-	-	171	400.8	-	-	-	-	-14	-1.8
21	561	404.2	-	-	617	398.5	610	354.8	583	369.8	-3.9	12.2	-	-	612	384.8	565	410.5	-	-	-0.7	4.8
22	984	395.8	-	-	1010	347	994	335.2	991	317.7	-0.7	19.7	-	-	1002	414.4	-	-	-	-	-1.8	-4.7
23	121	385.6	-	-	140	337	121	316.6	111	309.2	8.3	19.8	-	-	138	359.8	130	358.8	-	-	-7.4	6.9

2.5.5. Conclusiones del uso de la *BDP* para el problema *MMSP-W* cuando se incorporan restricciones en el mix de producción

En este caso, se incorporó al procedimiento *BDP* (aquí llamado *BDP-2*, con dos versiones) usado para resolver el problema nuevas características deseables en el caso de una línea de montaje de productos mixtos, como es el caso de la regularidad de la producción, un objetivo deseable vinculado al problema *PRV*. Para incorporar esta nueva característica, el procedimiento usa cotas globales basadas en programación lineal, más en concreto, un modelo matemático que minimiza la sobrecarga de trabajo dada una subsecuencia de opereraciones en cualquier instante t. Dicho modelo matemático se amplia en cada una de las etapas con el fin de acelerar su resolución. Además, se ha incorporado al procedimiento nuevas reglas para descartar vértices basadas en la regularidad de la producción, derivadas de las propiedades de estas, así como dos nuevas pseudo-dominancias para reducir el espacio de búsqueda.

Para comprobar la calidad de ambos procedimientos (cada uno de ellos basado en un tipo de pseudo-dominancias) se han realizado dos experiencias computacionales. En la primera de ellas, con 225 instancias de la literatura, los procedimientos propuestos basados en *BDP-2* mejoraron los tiempos de CPU promedios del solver Gurobi y otros procedimientos basados en *BDP*. Sin embargo, al incorporar las restricciones en el mix de producción, obtuvimos mejoras en la regularidad de la producción ($\triangle_Q(X)$), empeorando ligeramente la sobrecarga W. Sin embargo, creemos que normalmente un ligero incremento de la sobrecarga compensa con creces la gran mejora que se obtiene en regularidad de la producción, ya que reduciremos stocks y estaremos más alineados con los objetivos de las filosofías de trabajo *JIT* y *DS*.

En la segunda experiencia computacional corresponde a un caso de estudio de la planta de motores de Nissan en Barcelona. En este caso, se han empleado 23 instancias correspondientes a un día completo de trabajo (con dos turnos, *Bloque I* de las instancias) y diferentes planes de demandas para cada uno de los 9 tipos de motores. La incorporación al procedimiento *BDP* original de los nuevos mecanismos, dando origen a *BDP-2*, permitieron que el procedimiento obtuviera buenos resultados en un tiempo reducido, mejorando los tiempos de CPU que necesita Gurobi y procedimientos *BDP* anteriores, que no eran tan rápidos como el nuevo procedimiento *BDP-2*. Además, el procedimiento *BDP-2* mejoró los valores para W y $\triangle_Q(X)$ respecto a Gurobi en casi todas las instancias.

Finalmente, entre ambas variantes del procedimiento propuesto, *BDP-2/1* y *BDP-2/2*, cada una de ellas basadas en unas pseudo-dominancias diferentes, la que mejor se comportó fue *BDP-2/1*, basada en las pseudo-dominancias *PSD_1*.

2.6. Comprobación de la calidad del procedimiento *BDP*: Algoritmos *GRASP* para el *MMSP-W* considerando libre interrupción de operaciones

Para contrastar la calidad del procedimiento *BDP* con otros procedimientos metaheurísticos, se ha diseñado un procedimiento *GRASP extendido* (*GRASP-x*) que admite diversas variantes en función de los valores asignados a tres parámetros. Entre dichas variantes se encuentran las heurísticas *Greedy* constructivas con posterior optimización local, los procedimientos *Multistart*, los algoritmos *GRASP* clásicos (Feo and Resende, 1995; Resende and Ribeiro, 2003) y los algoritmos *GRASP* en los que la probabilidad de selección de los candidatos se hace depender de la aptitud de éstos, la cual puede medirse a través de una función de calidad dependiente de cada candidato. Una extensa recopilación de trabajos sobre aplicaciones de éstos algoritmos se puede encontrar en (Festa and Resende, 2009a; Festa and Resende, 2009b).

2.6.1. Cotas usadas para el procedimiento *GRASP*

De forma análoga al esquema de acotación presentado en el apartado 2.4, el esquema de acotación y evaluación de la calidad de las soluciones ha sido idéntico al presentado en el subapartado 2.4.2 de dicho apartado.

Dicho esquema está basado en fijar los valores de las variables binarias $x_{i,\tau}$ de acuerdo con la subsecuencia parcial asociada a la etapa t ($\tau = 1, ..., t$) y en la relajación de las variables binarias a partir de la etapa actual ($\tau = t+1, ..., T$). El esquema de acotación se puede ver gráficamente en la Figura 2.10 de dicho apartado.

A modo de resumen, se vuelve a mostrar el modelo resultante de dichas imposiciones sobre las variables binarias, al que se denominó LB_M4:

$$Min \quad LB(W(\pi(t))) = \sum_{k=1}^{|K|} \left(b_k \sum_{t=1}^{T} w_{k,t} \right)$$

Sujeto a:

$$(2.30) - (2.37)\ y\ (2.39)\ de\ M4$$

$$x_{\pi_\tau,\tau} = 1 \qquad \forall \tau = 1, ..., t$$

$$0 \leqslant x_{i,\tau} \leqslant 1 \qquad \forall i = 1, ..., |I|; \forall \tau = t+1, ..., T$$

El programa lineal anterior nos ofrecerá una cota global de la sobrecarga total ($LB(W(\pi(t)))$), así como un valor de la sobrecarga asociada al segmento $\pi(t)$ (o sea, $W(\pi(t))$) y una cota de la sobrecarga asociada al complemento $R(\pi(t))$ (es decir, $LB(R(\pi(t)))$), tal y como se mostró en las ecuaciones (2.96) y (2.97). Además, también es posible obtener a partir de LB_M4 los instantes

relativos de finalización ($\hat{e}_{k,t}$) de la última operación de la secuencia parcial $\pi(t)$, en cada estación de trabajo de la forma: $\hat{e}_{k,t} = \hat{s}_{k,t} + \rho_{k,t} - w_{k,t}$, $\forall k$.

2.6.2. Algoritmos *GRASP* para el *MMSP-W*

Esta sección presenta los elementos básicos del procedimiento *GRASP* aplicado a la resolución del *MMSP-W* con estaciones de trabajo en serie y con libre interrupción de las operaciones.

Preliminares

La complejidad del *MMSP-W* con vínculos entre estaciones de trabajo y el interés de obtener soluciones para ejemplares del problema con dimensiones industriales, hace recomendable el uso de procedimientos heurísticos capaces de ofrecer soluciones aceptables con bajo esfuerzo de computación.
La metaheurística *GRASP* (Feo and Resende, 1995; Resende and Ribeiro, 2003) es de tipo multiarranque y está provista de 2 fases en cada iteración:

1. Un procedimiento *Greedy* que sirve para construir una solución aceptable sin que sea preciso alcanzar el óptimo global.

2. Una segunda fase para obtener un óptimo local dentro de un vecindario y teniendo como punto de partida la solución que resulta al aplicar el procedimiento *Greedy* de la fase_1. Obviamente, la solución ofrecida por el *GRASP* es la mejor entre las obtenidas en el conjunto de iteraciones.

Para la primera fase *Greedy* es importante definir un buen procedimiento que pueda ofrecer soluciones aceptables y una diversidad suficiente de soluciones que permitan explorar diferentes regiones en el espacio de soluciones. Para garantizar dicha diversidad se emplea el azar, de manera que, el siguiente elemento a añadir a una solución parcial se sortea entre los elementos de una lista restringida de candidatos (*RCL*); dicha lista contiene los candidatos que presentan los mejores valores en relación a una función (por ejemplo la cota descrita en el apartado anterior) diseñada para la selección.
Para resolver un problema de optimización mediante un procedimiento *GRASP* es preciso definir los siguientes elementos:

1. El proceso aleatorio empleado en la selección entre candidatos y el procedimiento *Greedy*.

2. El vecindario de una solución y, lógicamente, el procedimiento para explorar dicho vecindario.

3. El criterio de finalización del algoritmo, normalmente vinculado a un máximo número de iteraciones o al tiempo de ejecución.

En el algoritmo 6 se presenta un esquema general del algoritmo *GRASP*:

Algoritmo 6 *Esquema general de la metaheurística* GRASP.

1: Inicialización
2: **para** $iter = 1$ **to** $iter_{max}$ **hacer**
3: Solución ⟵ Fase_constructiva (*Semilla*)
4: Solución ⟵ Mejora_local (*Solución*)
5: Actualizar_Solución (*Solución, Mejor_solución*)
6: **fin para**
7: Salida: *Mejor_solución*

En algunos casos, al procedimiento *GRASP*, se le puede añadir un post-proceso que permita combinar las soluciones generadas (Laguna and Martí, 1999) o también un mecanismo de auto-adaptación de los parámetros del algoritmo durante el proceso de búsqueda (Prais and Ribeiro, 2000).

Fase Greedy de construcción de una solución

El procedimiento implementado para esta fase del algoritmo (ver algoritmo 7) construye progresivamente una secuencia seleccionando, en cada etapa asociada con el instante, $t = 1, ..., T$, un elemento candidato a partir de una lista restringida de éstos (*RCL*).
En efecto, llegados a la etapa t, en la que se dispone de una secuencia (*solución*) parcial $\pi(t)$, para cada producto i con demanda pendiente aún, se determina el índice f_i $(\forall i : x_i < d_i)$ a partir del valor de la cota $LB(W(\pi(t) \cup \{i\}))$; tras ello, la lista de candidatos se construye en 2 pasos:

1. En el primero, a partir del parámetro $Z \in [0, 1]$ denominado impedancia, se seleccionan todos los productos con un valor de cota no superior a $1/Z$ veces el valor de la menor de ellas (cota correspondiente al mejor candidato).

2. En el segundo paso, se seleccionan como máximo los L mejores candidatos (ordenados por cota, de menor a mayor) incluyendo en la lista, claro está, los productos empatados en cota con el *L-ésimo* candidato.

La selección del tipo de producto a secuenciar se realiza aleatoriamente con una probabilidad de selección, g_i, que se hace depender de la calidad de las soluciones parciales (medida a través del índice f_i). Además, se ha incorporado un mecanismo para modular dichas probabilidades mediante la utilización de dos parámetros:

1. El coeficiente de elasticidad aditiva, f_0, que puede corresponderse con el valor de una solución inicial y permite la concentración relativa de los valores de la aptitud de los productos candidatos.

2. El coeficiente de elasticidad potencial, η, que sirve para concentrar ($\eta \in [0,1]$) o dispersar ($\eta > 1$) los valores de la aptitud en términos absolutos.

Algoritmo 7 *Fase constructiva GRASP para el MMSP-W*

Entrada: T, I, K, $d_i(\forall i)$, l_k, $b_k(\forall k)$, $p_{i,k}(\forall i, \forall k)$, c, Z, L, f_0, η

Paso 0 - Inicialización:
$t = 0$
$x_i = 0$ $\{\forall i \in I$, siendo x_i el número de unidades de tipo i secuenciadas hasta el instante $t\}$
$\pi(t) = \{\varnothing\}$ {siendo $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$ la secuencia parcial construida hasta el instante t}
Paso 1 - Cálculo del índice f:
$\forall i \in I$: $x_i < d_i$, determinar: $f_i = LB^*(W(\pi(t) \cup \{i\}))$ {siendo $LB^*(W(\pi(t) \cup \{i\}))$ el valor de la solución óptima ofrecida por el programa lineal *LB_M4* dada la secuencia parcial $\pi(t) \cup \{i\}$.}
Paso 2 - Creación de la lista de candidatos *RCL*:
Sea $f^* = \min_{x_i < d_i}\{f_i\}$; $RCL(f) = \{i \in I : (x_i < d_i) \wedge (f_i \leq \min\{f_0, f^*/Z\}\}$ {donde $Z \in [0,1]$ es la impedancia sobre el conjunto de elementos compatibles y f_0 es la elasticidad aditiva que que se puede corresponder con el valor de una solución de referencia}
si $|RCL(f)| \leq L$ **entonces**
    $RCL = RCL(f)$
si no, si $|RCL(f)| > L$ **entonces**
    Sea $i_L \in RCL(f)$ el tipo de producto que ocupa la *L-ésima* posición en la lista $RCL(f)$ ordenada no-decrecientemente respecto al índice f. Hacer: $RCL = \{i \in RCL(f) : f_i \leq f_L\}$
fin si
Paso 3 - Selección del tipo de producto a secuenciar:
$\forall i \in RCL$, determinar: $g_i = \frac{(f_0 - f_i)^\eta}{\sum_{j \in RCL}(f_0 - f_j)^\eta}$ { donde η es la elasticidad potencial.}
Seleccionar, por sorteo, con probabilidades g_i $\forall i \in RCL$ un tipo de producto; sea i^* el resultado de esta selección.
Paso 4 - Actualización:
$x_{i^*} \leftarrow x_{i^*} + 1$; $t \leftarrow t + 1$; $\pi_t = i^*$
Paso 5 - Finalización:
si $t < T = \sum_{i \in I} d_i$ **entonces**
    Ir a **Paso 1**
si no
    Finalizar
fin si

Nótese que la formulación propuesta en el algoritmo 7 comprende como casos particulares los siguientes procedimientos:

1. Los algoritmos *GRASP* tradicionales con tratamiento de empates incorporado, pues basta hacer $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L menor que el número de candidatos.

2. *Multistart*, haciendo $Z \to 0$, $f_0 \to \infty$, $\eta = 1$ y fijar un valor de L igual al número de candidatos.

3. Las heurísticas *Greedy* con tratamiento de empates con $Z = 1$. Nótese además que si $Z \to 0$ y L es suficientemente grande ($L = |I|$) todos los elementos compatibles son candidatos y toda solución del *MMSP-W* tiene una probabilidad no nula de ser generada por el procedimiento.

Fase de mejora local

Se propone una mejora local exhaustiva tipo 2-intercambio entre dos elementos de la secuencia en curso de mejora; el intercambio tentativo se produce, lógicamente, cuando dichos elementos corresponden a tipos de productos distintos en el problema *MMSP-W*. La exploración del vecindario es determinista y se realiza de izquierda a derecha.
A partir de una secuencia en curso, $\pi_c(T)$, con valor $LB^*(W(\pi_c(T))$ obtenido a partir del programa lineal *LB_M4*, se genera una secuencia vecina, $\pi_v(T)$, mediante un 2-intercambio tentativo entre los elementos que ocupan las posiciones t y t' de la secuencia en curso. Si $LB^*(W(\pi_v(T)) < LB^*(W(\pi_c(T))$, el intercambio se consolida, $\pi_v(T)$ se convierte en la nueva secuencia en curso y se reinicia el proceso de intercambios. En caso contrario, se prosigue con la generación de una nueva secuencia tentativa, una secuencia vecina, mediante otro 2-intercambio tentativo. El procedimiento finaliza cuando ninguna solución vecina tiene un valor de sobrecarga menor que el de la solución en curso.

2.6.3. Experiencia computacional

Se ha realizado una experiencia computacional que emplea los 225 ejemplares del problema *MMSP-W* recogidos en apartados anteriores (ver (Bautista and Cano, 2008; Bautista and Cano, 2011)). Dichos ejemplares se construyen a partir de 45 programas de producción y 5 estructuras de tiempos de proceso de las operaciones. Todos los ejemplares presentan 4 tipos de productos distintos ($|I| = 4$) y 4 estaciones de trabajo en serie ($|K| = 4$). Las soluciones óptimas de los 225 ejemplares se han obtenido empleando el solver *CPLEX v11.0* (*para un procesador*). Dichas soluciones se comparan con los resultados ofrecidos por 7 algoritmos derivados del procedimiento general *GRASP-x* para el que se han fijado los valores de los parámetros Z, L, f_0 y η (ver Tabla 2.21), siendo f_{BDP} el valor de referencia, para cada ejemplar, ofrecido por el procedimiento basado en la programación dinámica acotada descrito en el apartado 2.3.
Los algoritmos resultantes tras asignar valores a los 4 parámetros son:

- (*G*) una heurística *Greedy* constructiva con posterior mejora local.
- (*M*) un procedimiento *Multistart* tradicional con mejora local.
- (*GR-01/2*) un algoritmo *GRASP* tradicional con una lista *RCL* limitada a 2 candidatos.
- (*GR-5/2*) un algoritmo *GRASP* con lista limitada a 2 candidatos y con probabilidades de selección de éstos levemente dependientes de su aptitud.

- (*GR-9/2*) un algoritmo *GRASP* con alta dependencia entre las probabilidades de selección de los candidatos y su aptitud y con lista restringida a 2 candidatos
- Dos algoritmos *Multistart*, (*GR-5/4*) y (*GR-9/4*) con probabilidades de selección de los candidatos idénticas a (*GR-5/2*) y (*GR-9/2*), respectivamente

En todos los algoritmos se fija el número máximo de iteraciones ($iter_{max} = 10$), impuesto a cada uno de los 7 algoritmos en el proceso de resolución de cada ejemplar. Para obtener los valores de las soluciones óptimas de *LB_M4* y calcular el índice f (ver algoritmo 7, Paso 1), se ha empleado el solver Gurobi v4.5.0.

Tabla 2.21: Algoritmos derivados de *GRASP-x*. Características.

Alg.	Z	L	f_0	η
G	1	1	f_{BDP}	1
M	0.01	4	f_{BDP}/Z	1
GR-01/2	0.01	2	f_{BDP}/Z	1
GR-5/2	0.5	2	f_{BDP}/Z	1
GR-9/2	0.9	2	f_{BDP}/Z	1
GR-5/4	0.5	4	f_{BDP}/Z	1
GR-9/4	0.9	4	f_{BDP}/Z	1

Los procedimientos han sido programados en gcc v. 4.2.1, en un ordenador Macintosh iMac con un procesador Intel Core i7, 2.93 Ghz. y 8 Gb de memoria RAM, usando MAC OS X 10.6.8 como sistema operativo. Ni la implementación ni el compilador hacen uso de *threads* ni de otra forma de código paralelo, y, por tanto, el ordenador actúa como un único procesador a 2.93 GHz.
Los resultados del experimento se resumen en las Tablas 2.22, 2.23 y 2.24. En la Tabla 2.22 se recoge:

- El número de óptimos alcanzado (*#opt*) por cada uno de los 7 algoritmos sobre los 225 ejemplares del *MMSP-W*.
- El promedio de la *desviación porcentual relativa* (RPD=((*Solución - Óptimo*)/*Óptimo*)×100), para los ejemplares y cada algoritmo, tanto para la fase_1 *Greedy* constructiva del *GRASP* ($\overline{RPD_1}$) como para el proceso completo ($\overline{RPD_2}$) que incluye el procedimiento de mejora local.
- El tiempo medio de *CPU*, por ejemplar, requerido por una iteración de cada uno de los 7 algoritmos *GRASP* ($\overline{CPU}$).

La Tabla 2.23 muestra, para cada uno de los 5 bloques de ejemplares formados a partir de familias de los 45 programas de producción, los valores de $\overline{RPD_2}$, promediados por bloque, que ofrece

cada uno de los 7 algoritmos. En la Tabla 2.24 se recogen, para cada una de las 5 estructuras de tiempos de proceso de las operaciones, los valores de $\overline{RPD_2}$, promediados por estructura, que ofrece cada uno de los 7 algoritmos.

Tabla 2.22: Número de óptimos, $\overline{RPD_1}$, $\overline{RPD_2}$ y $\overline{CPU}$ obtenidos a partir de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$
G	103	9.92	2.46	0.29
M	151	28.81	1.06	0.30
GR-01/2	*159*	15.71	*0.87*	0.31
GR-5/2	150	14.34	1.00	0.31
GR-9/2	155	*7.67*	0.95	0.28
GR-5/4	155	24.89	0.89	0.31
GR-9/4	152	8.68	0.91	0.27

Tabla 2.23: $\overline{RPD_2}$ obtenida por los 7 algoritmos para cada bloque de programas de producción

	B1	*B2*	*B3*	*B4*	*B5*
G	0.37	2.02	4.02	2.47	2.46
M	0.22	0.65	2.37	1.02	0.93
GR-01/2	0.04	*0.54*	*1.61*	0.74	0.91
GR-5/2	0.04	0.89	2.53	*0.55*	0.81
GR-9/2	*0.00*	0.71	2.04	0.72	0.89
GR-5/4	0.04	0.87	1.93	0.56	*0.79*
GR-9/4	*0.00*	0.70	1.85	0.90	0.85

En la Tabla 2.22 se observa que el algoritmo que obtuvo peores resultados en cuanto a óptimos alcanzados es el *G* (*103 óptimos sobre 225 ejemplares*), mientras que *GR-01/2* fue la heurística con mayor número de éxitos (*159 sobre 225*). En cuanto a la calidad de las soluciones, medida a través de $\overline{RPD_2}$, se puede observar que el comportamiento de todos los algoritmos, exceptuando el *G*, fue similar, con unos valores de $\overline{RPD_2}$ alrededor del 1 %, e inferiores a la mitad del valor correspondiente al algoritmo *G*. En cuanto a los valores de $\overline{RPD_1}$ correspondientes a la fase constructiva de los algoritmos, se puede observar que *GR-9/2* es el que ofrece mejor comportamiento, seguido de muy cerca por *GR-9/4* y *G*; en este caso, el algoritmo constructivo que dio peores resultados es *M*. Por otra parte, los tiempos de *CPU* requeridos por cada ejemplar en cada iteración son si-

Tabla 2.24: $\overline{RPD_2}$ obtenida por los 7 algoritmos para cada estructura de tiempos de proceso de las operaciones

	E1	*E2*	*E3*	*E4*	*E5*
G	4.18	3.26	1.67	0.08	3.10
M	2.19	*1.07*	0.53	*0.00*	1.52
GR-01/2	1.79	1.18	*0.33*	*0.00*	1.07
GR-5/2	2.32	1.10	0.42	*0.00*	1.16
GR-9/2	1.87	1.18	0.42	*0.00*	1.26
GR-5/4	1.76	1.29	0.43	*0.00*	*0.98*
GR-9/4	*1.74*	1.31	0.35	*0.00*	1.17

milares para todos los procedimientos (*alrededor de 0.30s*). Dichos tiempos son muy inferiores a los empleados en promedio por el solver *CPLEX* y ligeramente inferiores a los del procedimiento *BDP* (recogidos en el apartado 2.4) que son del orden de 81.7s y 3.3s, respectivamente.
Examinando los resultados recogidos en la Tabla 2.23 podemos concluir que el peor procedimiento para todos los bloques fue *G* (*en sintonía con los valores de* $\overline{RPD_2}$ *para los 225 ejemplares*); el resto de algoritmos se comporta de forma irregular en todos los bloques, aunque ofrecen mejores resultados que *G*. En cuanto a resultados por bloques, las mejores soluciones se concentraron en el bloque 1, mientras que las peores soluciones lo hicieron en el bloque 3.
Si observamos los resultados por estructuras (*tabla 2.24*), la estructura, sobre la que se obtuvieron mejores resultados, fue la 4; siendo la estructura 1 la más difícil de resolver para todos los algoritmos. De nuevo, el algoritmo que presentó peores promedios en este caso fue *G*, comportándose el resto de algoritmos de una forma similar.
Si comparamos los mejores resultados obtenidos usando los algoritmos *GRASP* con los resultados obtenidos por el procedimiento *BDP* presentado en el apartado 2.4, podemos observar:

- Respecto al número de óptimos alcanzados, el procedimiento *BDP* alcanza un mayor número de óptimos, 213 respecto a 155 de los mejores procedimientos *GRASP*.
- Respecto a la *RPD* tomando como referencia el valor óptimo obtenido con CPLEX, el procedimiento *BDP* tiene una *RPD* promedio para los 225 ejemplares de 0.03 %. En cambio, los procedimientos basados en *GRASP* tienen su mejor *RPD* promedio ($\overline{RPD_2}$) en 0.87 %.
- En términos de estructuras, los procedimientos *GRASP* presentaron un peor valor en $\overline{RPD_2}$ en las estructuras 1, 2, 3 y 5, donde usando *BDP* se obtuvieron todos los óptimos para las estructuras 1 y 2, y una *RPD* de 0.11 % y 0.06 % para las estructuras 3 y 5. En cambio, el mejor *GRASP* sólo fue capaz de alcanzar un 1.74 %, 1.07 %, 0.33 % y 0.98 %, para las estructuras 1,

2, 3 y 4, respectivamente. Sin embargo, esto no ocurrió en la estructura 4, donde *BDP* obtuvo una *RPD* promedio de 0.10 %, mientras que los algoritmos *GRASP* alcanzaron todos los óptimos. Esto puede ser causado por la diferente estrategia de cada uno de los algoritmos, favoreciendo en este caso a los procedimientos *GRASP* el alcanzar las mejores soluciones en esta estructura en particular.

En base a los resultados obtenidos, los procedimientos *GRASP* se han mostrado más rápidos en términos de CPU promedio, pero el procedimiento *BDP* presentado en el apartado 2.4 es más competitivo en calidad de las soluciones.

2.6.4. Conclusiones del empleo del *GRASP* en el *MMSP-W*

La propuesta *GRASP* se concreta en 7 algoritmos que se aplican a la variante del *MMSP-W*, objeto de estudio. Bajo la capa de la fase constructiva de los algoritmos maestros, se determinan los valores de la función de aptitud correspondientes a la incorporación, a la secuencia parcial en curso, de cada uno de los tipos de producto candidatos. La determinación de dichos valores, a través del solver Gurobi, se realiza mediante un programa lineal asociado al problema *MMSP-W*, relajado en las condiciones binarias para las variables de secuenciación.
La experiencia computacional sobre 225 ejemplares pone de manifiesto la competencia de *GRASP* (en tiempos de computación) frente a procedimientos exactos de resolución basados en la programación lineal entera mixta (tales como los incorporados al solver CPLEX) y un peor comportamiento o calidad de las soluciones ante el uso de la programación dinámica acotada (*BDP*).

Capítulo 3

El *BFSP*

3.1. Introducción y estado del arte

Aunque el problema del taller mecánico admite muchísimas variantes, dependiendo del flujo de las piezas en el taller y el índice para medir la eficiencia que se escoja, el problema *FSP* (Flowshop Scheduling Problem) es uno de los problemas que ha recibido más atención en los últimos 50 años y continuamente recibe la atención de profesionales e investigadores debido a la gran variedad de contextos productivos que permite modelizar.
En el *FSP*, un conjunto n de piezas deben ser procesados en un conjunto de m máquinas. Todas las piezas deben ser procesados por todas las máquinas en el mismo orden, empezando en la máquina 1 y finalizando en la máquina m. Cada pieza $i \in I$, requiere un tiempo de proceso $p_{i,k} > 0$, en cada una de las máquinas $k \in K$. El objetivo es encontrar una secuencia de las piezas que minimice el instante de finalización en que todas las piezas en el taller han sido procesadas.
En la versión más popular del problema, conocida como *PFSP* (Permutation Flowshop Scheduling Problem), los espacios de almacenamiento o *buffers* entre dos fases consecutivas del proceso son ilimitados. Sin embargo, existen sistemas productivos en diversos sectores en los cuales la capacidad de almacenamiento es limitada o nula. Si asumimos que no hay posibilidad de almacenar las piezas entre dos fases consecutivas del proceso, es necesario un cambio estructural en el comportamiento del sistema, ya que una pieza no puede abandonar la máquina que la está procesando hasta que la siguiente máquina esté libre. Si la máquina no esta libre, la pieza ha de permanecer en la máquina previa, bloqueándola y no permitiendo que comience el proceso en la siguiente pieza. Esta variante se conoce como *BFSP* (Blocking Flowshop Scheduling Problem), en la cual los espacios de almacenamiento son tratados como máquinas *dummy* con un tiempo de proceso igual a 0 (McCormick, Pinedo, Shenker and Wolf, 1989). Utilizando la nomenclatura de (Graham, Lawler, Lenstra and Rinnooy Kan, 1979), el problema considerado se denomina como $Fm|block|C_{max}$ (y el *PFSP* por $Fm|prmu|C_{max}$).
Parte del trabajo presentado en este capítulo se puede encontrar en (Bautista, Cano, Companys

and Ribas, 2012; Bautista, Cano, Alfaro and Batalla, 2013).

3.1.1. Estado del arte

En (Hall and Sriskandarajah, 1996), encontramos una revisión de la literatura sobre *flow shop* con bloqueo y sin esperas en el proceso. En este problema, existen casos particulares en los cuales se puede transformar en otros problemas. Por ejemplo, en el caso que el número de máquinas sea dos, es posible reducir el problema $F2|block|C_{max}$ a un *TSP* (problema del viajante de comercio) tal y como se muestra en (Reddi and Ramamoorthy, 1972). Este *TSP* tiene $n + 1$ ciudades $(0, 1, 2, ..., n)$ y debido a que el *TSP* ha sido ampliamente tratado en la literatura científica, encontramos métodos de resolución en tiempo polinomial usando el algoritmo propuesto en (Gilmore and Gomory, 1964) y (Gilmore, Lawler and Shmoys, 1991). En este caso, la secuencia de ciudades en un camino óptimo corresponde a una permutación óptima para el problema original.
En cuanto a la dificultad computacional del problema Blocking Flow Shop, (Hall and Sriskandarajah, 1996) demostraron, usando resultados de (Papadimitriou and Kanellakis, 1980), que el problema $Fm|block|C_{max}$ para $m \geqslant 3$ máquinas es *NP-hard*.
Los diferentes autores de la literatura han usado diferentes acercamientos para la resolución del problema. Por ejemplo, (Ronconi, 2005) presentó un algoritmo *branch-and-bound* que usaba una cota inferior muy elaborada. Otras referencias como (Companys and Mateo, 2007) utilizaron la propiedad de reversibilidad del problema, usando un algoritmo *branch-and-bound* doble. (Ronconi, 2005) y (Companys and Mateo, 2007) utilizaron para aportar soluciones el conjunto de ejemplares de Taillard (Taillard, 1993).
Debido al hecho de que el problema es *NP-hard*, en la literatura se encuentran varios procedimientos heurísticos para el problema $Fm|block|C_{max}$. A continuación se recogen algunos ejemplos de las heurísticas desarrolladas en la literatura:

- (McCormick et al., 1989) estudiaron un caso especial del problema y propusieron una heurística constructiva para resolverlo.
- (Leisten, 1990) adaptó algunos de los procedimientos propuestos para el *PFSP* y concluyó que la heurística *NEH*, propuesta por (Nawaz, Enscore and Ham, 1983), tenía un buen comportamiento para el problema *BFSP*.
- (Caraffa, Ianes, Bagchi T and Sriskandarajah, 2001) propusieron un algoritmo genético (*GA*).
- (Ronconi, 2004) propuso cuatro procedimientos heurísticos, dos de ellos basados en *NEH*. Posteriormente desarrollo un procedimiento basado en *branch-and-bound* en (Ronconi, 2005).
- (Wang, Zhang and Zheng, 2006) propusieron un algoritmo genético híbrido (*HGA*).
- (Grabowski and Pempera, 2007) presentaron dos algoritmos basados en búsqueda tabú (*TS*) y los evaluaron usando las soluciones obtenidas por (Ronconi, 2005).

- (Liu, Wang and Jin, 2008) presentaron un algoritmo basado en optimización de enjambres (*HPSO*).
- (Qian, Wang, Huang and Wang, 2009) propusieron un algoritmo basado en evolución diferencial (*DE*).
- Más recientemente, (Wang, Pan, Suganthan, Wang and Wang, 2010) propusieron un algoritmo híbrido basado en evolución diferencial discreta (*HDDE*) y compararon sus resultados con los obtenidos por la búsqueda tabú propuesta en (Grabowski and Pempera, 2007).
- Uno de los trabajos más recientes, (Ribas et al., 2011), propuso un algoritmo *Greedy* iterado (*IGA*), el cual mejoró los resultados de (Wang et al., 2010). Además, esta referencia contiene una lista actualizada con las mejores soluciones conocidas para los ejemplares de Taillard.

3.2. Formalización del problema

En el instante 0, se deben procesar n piezas en el mismo orden en cada una de las m máquinas. Cada pieza ha de ser procesada desde la máquina 1 hasta la máquina m. El tiempo de proceso para cada operación es $p_{i,k}$, donde $k \in K = \{1, 2, \ldots, m\}$ indica una máquina y $i \in I = \{1, 2, \ldots, n\}$ una pieza. Los tiempos de preparación están incluidos en los tiempos de proceso. Estos tiempos están fijados a priori y deben ser positivos.
En el caso que exista suficiente espacio de almacenamiento entre la máquina k y la máquina $k+1$, la pieza i puede esperar en ese espacio a que se libere la próxima máquina ($k+1$). Como consecuencia de esto la máquina k queda liberada y puede trabajar en otra pieza (caso del *PFSP*).
Sin embargo, si no existe almacenamiento entre etapas, las colas intermedias de piezas esperando en el sistema para continuar su operación no están permitidas. Esto quiere decir que si la operación en la máquina k para la pieza i se ha completado y la próxima máquina, $k+1$, está todavía ocupada en completar la pieza previa, la pieza completada i tiene que ser bloqueada en la máquina k. Este bloqueo que ocurre en este caso es el que da nombre al problema *BFSP*.
La función objetivo que se considera es la minimización del instante de finalización de la última pieza del taller, también conocido en la literatura como *makespan* (C_{max}).
Dada una permutación, π, de las n piezas, $[t]$ indica la pieza que ocupa la posición t en la secuencia. Por ejemplo, en $\pi = (3, 1, 2), [1] = 3, [2] = 1, [3] = 2$. Para esta permutación, en cada máquina, la pieza 2 ocupa la posición 3. Con el fin de formalizar el problema *PFSP*, definimos:

- $s_{k,t}$ corresponde al instante de inicio de la operación en la máquina k para la pieza que ocupa la posición t de la secuencia.
- $e_{k,t}$ corresponde al instante en que la pieza que ocupa la posición t concluye su operación en la máquina k.

- $p_{i,k}$ corresponde al tiempo de proceso para cada pieza (i) en cada máquina (k).
- C_{max} corresponde al instante de finalización de la última pieza del taller.

Teniendo en cuenta las definiciones anteriores definimos las restricciones que afectan al problema $Fm|prmu|C_{max}$ (*PFSP*) así:

$$s_{k,t} + p_{[t],k} \leqslant e_{k,t} \qquad k = 1, ..., m; t = 1, ..., n \tag{3.1}$$

$$s_{k,t} \geqslant e_{k,t-1} \qquad k = 1, ..., m; t = 1, ..., n \tag{3.2}$$

$$s_{k,t} \geqslant e_{k-1,t} \qquad k = 1, ..., m; t = 1, ..., n \tag{3.3}$$

$$C_{max} = e_{m,n} \tag{3.4}$$

Siendo $e_{k,0} = 0 \ \forall k, e_{0,t} = 0 \ \forall t$, las condiciones iniciales.
La programación es semiactiva si la ecuación (3.1) se escribe como $s_{k,t}+p_{[t],k} = e_{k,t}$ y las ecuaciones (3.2) y (3.3) se pueden resumir como $s_{k,t} = max\{e_{k,t-1}, e_{k-1,t}\}$.
Cuando no existe espacios de almacenamiento entre etapas (problema $Fm|block|C_{max}$, *BFSP*), si una pieza i finaliza su operación en la máquina k y la próxima máquina, $k+1$, está todavía ocupada con la pieza anterior, la pieza completada i debe permanecer en la máquina k, de forma que esa pieza estará bloqueada a la espera de que la máquina $k + 1$ sea liberada. Esta condición requiere una ecuación adicional (3.5) para completar las condiciones del problema:

$$e_{k,t} \geqslant e_{k+1,t-1} \qquad k = 1, ..., m; t = 1, ..., n \tag{3.5}$$

En este caso debemos tener en cuenta que debemos añadir la condición inicial $e_{m+1,t} = 0$, $t = 1, 2, ..., n$.
La programación obtenida es semiactiva si las ecuaciones (3.1) a (3.5) se resumen como (3.6):

$$e_{k,t} = max\{s_{k,t} + p_{[t],k}, e_{k+1,t-1}\} \qquad k = 1, ..., m; t = 1, ..., n \tag{3.6}$$

Como consecuencia, el problema $Fm|prmu|C_{max}$, o *PFSP*, puede ser visto como una relajación del problema $Fm|block|C_{max}$, o *BFSP*.

3.3. Un ejemplo para el *PFSP* y el *BFSP*

Para ilustrar el problema anterior, usaremos el siguiente ejemplo:
Existen 6 piezas ($n = 6$), los cuales son procesadas en 3 máquinas ($m = 3$). Los tiempos de proceso de cada pieza (A, B, C, D, E, F) en cada máquina (m_1, m_2, m_3) son los siguientes (Tabla 3.1):

Tabla 3.1: Tiempos de proceso ($p_{i,k}$) para cada pieza i, en cada máquina k

	A	B	C	D	E	F
m_1	1	4	9	10	3	1
m_2	3	5	9	2	5	4
m_3	10	9	6	3	4	1

En la Figura 3.1 y la Figura 3.2 se observa el diagrama asociado a la secuencia $A-B-C-D-E-F$ para el ejemplo, sin bloqueo y con bloqueo, respectivamente.

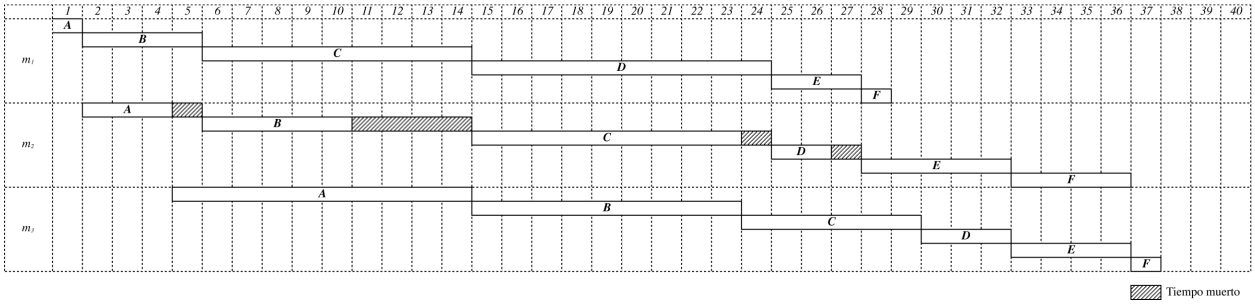

Figura 3.1: Diagrama de Gantt para el caso sin bloqueos. $C_{max} = 37$

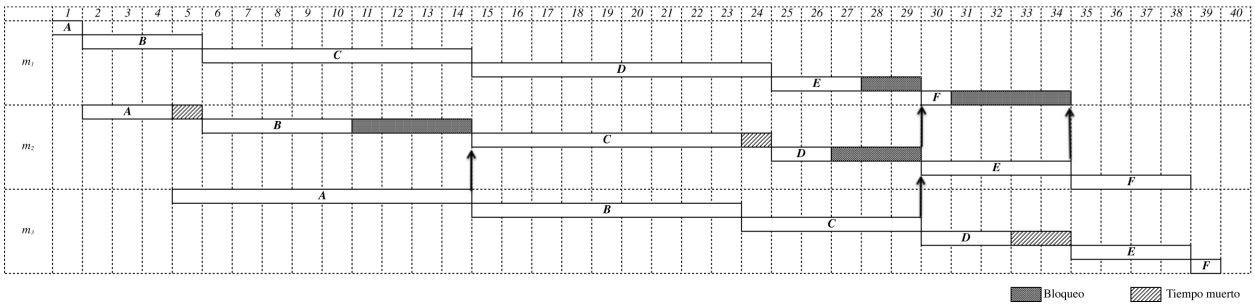

Figura 3.2: Diagrama de Gantt para el caso con bloqueos. $C_{max} = 39$

A la vista de la Figura 3.1, que corresponde a la resolución de un problema *PFSP*, la secuencia $A-B-C-D-E-F$ tiene asociada un $C_{max} = 37$. La trama rayada corresponde a los tiempos muertos asociados a la secuencia (definimos como tiempo muerto el tiempo en que la máquina podría estar produciendo, pero a consecuencia de que la pieza que debería ejecutarse todavía se está ejecutando en una máquina anterior, ésta se ha de esperar a que le llegue la pieza para ponerse a trabajar en ella). En algunos casos, puede ocurrir que la máquina actual haya finalizado una pieza, pero que la siguiente máquina todavía este ocupada con la pieza anterior. Esto ocasiona que la pieza tenga que ir a un buffer o almacenamiento intermedio. Este último caso ocurre en la

pieza B en m_2 en el periodo 11 hasta 14, que debe esperar a que la máquina m_3 esté libre. Lo mismo ocurre en la pieza D en m_2 en el periodo 27 a 29 y en la pieza F en m_1 en el periodo 29 a 32.
En el caso que no dispongamos *buffers* o almacenamientos intermedios (*BFSP*), caso que ilustra la Figura 3.2, se observa que la misma secuencia en este caso tiene asociada un $C_{max} = 39$ (además, para este ejemplo, la secuencia $A - B - C - D - E - F$ corresponde con la secuencia óptima). Esto es debido a que aparecen bloqueos (representados en trama sombreada y mostrando con una flecha la máquina que produce el bloqueo, en la Figura 3.2). Estos bloqueos aparecen en la pieza B en m_2 en el periodo 11 a 14, provocado por que la máquina m_3 no está disponible cuando m_2 acaba su operación en esa pieza (flecha en el instante 14) y por tanto debe bloquear a la espera de que la máquina m_3 finalice su operación. Lo mismo ocurre en m_2 en la pieza D en el periodo 27 a 29, en m_1 en la pieza E en el periodo 28 a 29 y en m_1 en la pieza F en el periodo 31 a 34. Estos bloqueos son los que producen el incremento de la C_{max} de 37 a 39, con lo cual el taller finalizará sus operaciones 2 unidades de tiempo más tarde.
Tal y como se observa con las diferencias presentes entre la Figura 3.1 y la Figura 3.2, los problemas *PFSP* y *BFSP* no son equivalentes, aunque en algunos casos (por ej. cuando no ocurren bloqueos) si que se puede dar este caso.

3.4. Grafo asociado al problema

De forma similar al capítulo anterior, y a (Bautista et al., 1996a) y (Bautista and Cano, 2011), es posible construir un grafo conexo sin bucles o ciclos directos de $T+1$ etapas. Como anteriormente, un vértice de la etapa t ($t = 0, ..., T$) se define como $J(t)$, estando dicho vértice asociado a una subsecuencia parcial de t piezas. Además se define $J(t, j)$ ($j = 1, ..., |J(t, j)|$) como un vértice j de la etapa t, el cual esta representado por la tupla $(\vec{q}(t, j), \vec{e}(t, j), C_{max}(t, j))$ donde:

- $\vec{q}(t, j) = (q_1(t, j), ..., q_n(t, j))$ es el vector de los trabajos secuenciados (o no) en el *j-ésimo* vértice de la etapa t, donde $q_i(t, j),\ \forall i \in I\ (i = 1, ..., n)$ es el *i-ésimo* componente del vector $\vec{q}(t, j)$ que adopta el valor 1 si la pieza i se ha secuenciado y 0 en cualquier otro caso.

- $\vec{e}(t, j) = (e_1(t, j), ..., e_m(t, j))$ es el vector de los instantes de finalización de la última pieza secuenciada en cada máquina.

- $C_{max}(t, j)$ es el instante de finalización de la última pieza secuenciada en el vértice j de la etapa t.

El vértice $J(j,j)$ presenta las siguientes propiedades:

$$\sum_{i=1}^{n} q_i(t,j) = t \tag{3.7}$$

$$q_i(t,j) \in \{0,1\} \qquad i = 1,...,n \tag{3.8}$$

$$C_{max}(t,j) = e_m(t,j) \tag{3.9}$$

Para representar un vértice $J(t,j)$ es suficiente entonces con la siguiente información:

$$J(t,j) = \{(t,j), \vec{q}(t,j), \vec{e}(t,j)\} \tag{3.10}$$

Además, a parte de las subsecuencias de t piezas que cumplan las propiedades (3.7) y (3.8), es posible reducir el cardinal de vértices de la etapa t, reduciendo el espacio de búsqueda, estableciendo las siguientes reglas de dominancia y equivalencia:

$$J(t,j) \prec J(t,j') \Leftrightarrow \left[\vec{q}(t,j) = \vec{q}(t,j')\right] \wedge \left[\vec{e}(t,j) < \vec{e}(t,j')\right] \tag{3.11}$$

$$J(t,j) \equiv J(t,j') \Leftrightarrow \left[\vec{q}(t,j) = \vec{q}(t,j')\right] \wedge \left[\vec{e}(t,j) = \vec{e}(t,j')\right] \tag{3.12}$$

Para realizar una transición entre un vértice $J(t,j)$ de la etapa t y un vértice $J(t+1,j_i)$ a través de la pieza i ($J(t,j) \xrightarrow{i} J(t+1,j_i)$, en caso que $\vec{q}(t,j) \prec \vec{q}(t+1,j_i)$), es necesario determinar $J(t+1,j_i) = \{(t+1,j_i), \vec{q}(t+1,j_i), \vec{e}(t+1,j_i)\}$ de la siguiente manera:

$$q_i(t+1,j_i) = 1 \tag{3.13}$$

$$q_h(t+1,j_i) = q_h(t,j) \qquad \forall h : h \neq i \in I \tag{3.14}$$

$$e_k(t+1,j_i) = max\{e_k(t,j) + p_{i,k}, e_{k-1}(t+1,j_i), e_{k+1}(t,j)\} \qquad \forall k \in K \tag{3.15}$$

donde $e_0(t+1,j_i) = 0$.

La Figura 3.3 se muestra un esquema del grafo de estados asociado al *BFSP*. Los vértices iniciales $J(t,j)$ y $J(t,j')$ de la etapa t son desarrollados secuenciando una pieza i y i', respectivamente. Este desarrollo resulta en un único vértice, $J(t+1,j_i)$, usando las reglas de dominancia y equivalencia (3.11) y (3.12). Llegados a este paso, es posible determinar las propiedades del vértice $J(t+1,j_i)$ de la etapa $t+1$. Finalmente, éste vértice es desarrollado secuenciando una pieza i'', dando como resultado el vértice $J(t+2,j_{i''})$ de la etapa $t+2$.

Además, es posible calcular la contribución a la C_{max} parcial generada en la transición de $J(t,j)$ a $J(t+1,j_i)$ a través de la incorporación de una pieza i de la siguiente forma:

$$a((t,j) \rightarrow (t+1,j_i) = e_m(t+1,j_i) - e_m(t,j) \tag{3.16}$$

Bajo estas condiciones, encontrar una solución que optimice la C_{max} total es equivalente a encontrar un camino óptimo desde el vértice $J(0)$ de la etapa 0 al conjunto de vértices $J(T)$ de la etapa T. De nuevo, este entorno es propicio para la utilización del procedimiento *BDP* debido a que en los problemas realistas de la industria tienen un gran número de piezas y máquinas y dicho procedimiento nos permite reducir el espacio de búsqueda.

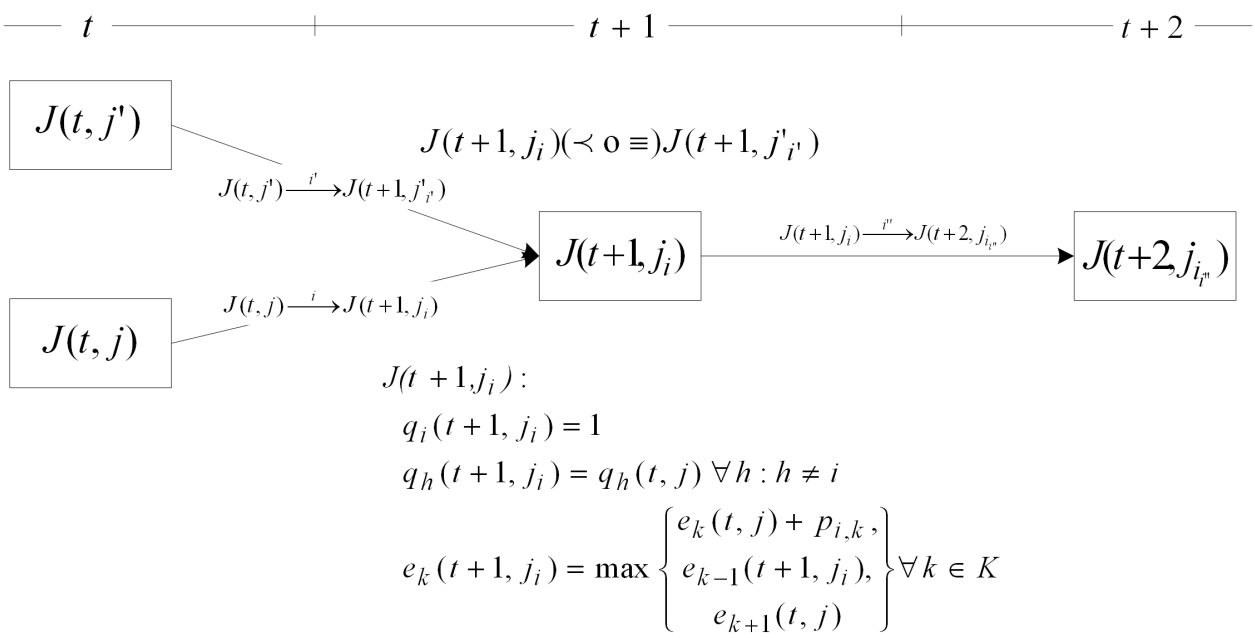

Figura 3.3: Esquema de transiciones a través del grafo de estados para el *BFSP*

3.5. Cotas para el problema

Uno de los inconvenientes del uso de la *BDP* es que este procedimiento necesita usar cotas asociadas al problema, con el fin de poder eliminar aquellas soluciones que a priori son menos prometedoras. En la metaheurística, la cota se utiliza para evaluar la calidad de la parte pendiente de secuenciar, como se explicará más adelante.
En este apartado, introduciremos la cota que usará nuestro procedimiento.
En nuestro caso, hemos escogido usar una cota asociada a las máquinas para el problema, basadas en las ofrecidas por (Lageweg, Lenstra and Rinnooy Kan, 1978) para el *PFSP*, adaptadas para el *BFSP*.
Sin embargo, existen cotas más refinadas, como por ejemplo las siguientes: la cota de Nabeshima (Bellman, Esogbue and Nabeshima, 1982), el uso de la resolución del problema 2 máquinas (ya que el caso 2 máquinas es un problema del tipo *P*, tal y como se menciona en (Companys, 2003)) como cota del problema, o incluso la resolución de un problema *TSP* (como se ha mencionado en el estado del arte sobre el problema), pero debido a los siguientes motivos, se decidió usar una cota más sencilla (como por ejemplo la cota asociada a las máquinas) con el fin de guiar el procedimiento *BDP*:

- La cota asociada a las máquinas es muy fácil de implementar y ofrece buenos resultados. El objetivo que se pretendía al usar esta cota era poder observar si el procedimiento daba

buenos resultados para el problema.

- Las otras cotas quizás pueden ser mucho más finas acotando el valor restante de las secuencias, pero debido a que son mucho más complejas de calcular (en tiempo de computación) y que las cotas en el procedimiento se calculan cada vez que se genera un nuevo vértice (y que el número de vértices es muy elevado), es probable que el tiempo de cálculo de resultados se incremente mucho.

3.5.1. Cotas globales para C_{max}

Si tenemos en cuenta a las máquinas de forma independiente, entonces es posible escribir:

$$LB1(k) = \sum_{i=1}^{n} p_{i,k} + \min_{(i,h)\in I: i\neq h} \left\{ \sum_{k'=1}^{k-1} p_{i,k'} + \sum_{k'=k+1}^{m} p_{h,k'} \right\} \qquad \forall k \in K \tag{3.17}$$

la cual es una cota para C_{max} a través de la máquina k.
Considerando todas las máquinas, tenemos:

$$LB1 = \max_{k\in K} \{LB1(k)\} \tag{3.18}$$

Siguiendo el mismo esquema, también es posible obtener una cota para C_{max} a través del trabajo i:

$$LB2(i) = \sum_{k=1}^{m} p_{i,k} + \sum_{h\in I: h\neq i} \min_{k\in K} \{p_{h,k}\} \qquad \forall i \in I \tag{3.19}$$

y considerando todos los trabajos:

$$LB2 = \max_{i\in I} \{LB2(i)\} \tag{3.20}$$

3.5.2. Cotas para C_{max} a través de un segmento dado

Asumamos que llevamos construida una subsecuencia de t piezas y que disponemos de la información $\vec{q}(t,j)$ y $\vec{e}(t,j)$. Para completar la secuencia de T unidades, necesitamos secuenciar $T-t$ piezas, cada una de ellas todavía no secuenciada. En este caso, es posible adaptar las cotas globales $LB1$ y $LB2$ tal y como sigue:

$$LB1(t,j) = \max_{k\in K} \left\{ e_k(t,j) + \sum_{i\in I: q_i(t,j)=0} p_{i,k} + \min_{i\in I: q_i(t,j)=0} \left\{ \sum_{k'=k+1}^{m} p_{i,k'} \right\} \right\} \tag{3.21}$$

$$LB2(t,j) = e_1(t,j) + \max_{i\in I: q_i(t,j)=0} \left\{ \sum_{k=1}^{m} p_{i,k} + \sum_{h\in I-\{i\}: q_h(t,j)=0} \min_{k\in K} \{p_{h,k}\} \right\} \tag{3.22}$$

3.6. El uso de la *BDP*

De forma similar al apartado 2.3.6 del capítulo anterior, se ha desarrollado un procedimiento basado en *BDP* utilizando el grafo y mecanismos de reducción de espacio de búsqueda presentados aquí, con algunas diferencias respecto al procedimiento descrito en dicho apartado:

- Por una parte, se ha fijado el número de transiciones desde un vértice de la etapa t al valor $n - t$.
- En este caso, se han desarrollado dos variantes basadas en *BDP*:
  1. El par ordenado de valores $(LB1(t,j), e_m(t,j))$ se usa como regla de prioridad o guía (GZ) del procedimiento. Una solución es más prometedora que otra cuando presenta una mejor cota para C_{max} ($LB1(t,j)$). En caso de empate entre dos soluciones parciales (igual valor para $LB1(t,j)$), la solución parcial con menor valor de $e_m(t,j)$ será considerada la mejor: $GZ = (LB1(t,j), e_m(t,j))$.
  2. En la segunda variante, el par ordenado de valores $(e_m(t,j), LB1(t,j))$ se usa como regla de prioridad o guia (GZ) del procedimiento. Una solución es más prometedora que otra cuando presenta un menor valor para su C_{max} parcial ($e_m(t,j)$). En caso de empate entre dos soluciones parciales (igual valor para $e_m(t,j)$), la solución parcial con menor valor de $LB1(t,j)$ será considerada la mejor: $GZ = (e_m(t,j), LB1(t,j))$.
- En este caso, LBZ es directamente la cota ofrecida por $LB1(t,j)$.

En tales condiciones, a continuación se resume el algoritmo *BDP* adaptado al problema (Variantes 1 y 2):

Algoritmo 8 *Esquema BDP para el problema* $Fm|block|C_{max}$

Entrada: $T, |I|, |K|, d_i(i = 1, ..., |I|), p_{i,k}(i = 1, ..., |I|; k = 1, ..., |K|), Z_0, H$

Salida: Lista de secuencias obtenidas con *BDP*

```
Inicialización: t = 0; LBZ_min = ∞
mientras t < T hacer
    t ← t + 1
    mientras lista de vértices consolidados en la etapa t − 1 (Λ(t − 1)) no vacía hacer
        Seleccionar_vértice(t)
        Desarrollar_vértice(t)
        Λ(t) ← Filtrar_vértices(Z_0, H, LBZ_min)
    fin mientras
    Finalizar_etapa()
fin mientras
```

En el procedimiento aparecen las siguientes funciones:

- Seleccionar_vértice(t): Se selecciona un vértice de la etapa $t-1$ en el orden establecido en la lista consolidada $\Lambda(t-1)$. Este orden depende de la variante del algoritmo usada: la Variante 1 ordena los vértices en orden no decreciente de $LB1$ y, para romper empates, en orden no decreciente de la C_{max} parcial de la subsecuencia del vértice; por otra parte, en la Variante 2 los vértices se ordenan en orden no decreciente de la C_{max} parcial, y en caso de empates, en orden no decreciente de $LB1$.

- Desarrollar_vértice(t): Desarrolla el vértice seleccionado en el paso anterior añadiendo un nuevo producto no secuenciado a la subsecuencia del vértice actual. Durante este desarrollo se calculan $LB1$, la C_{max} parcial y los instantes de finalización en cada una de las máquinas de la pieza añadida. Lógicamente, el desarrollo de un vértice implica evaluar todas las posibles extensiones de la subsecuencia del vértice actual, considerando una a una todas las piezas no secuenciadas.

- Filtrar_vértices(Z_0, H, LBZ_{min}): Escoge, de todos los vértices desarrollados en el paso anterior, un número máximo H de los vértices más prometedores. Los vértices escogidos son aquellos con mejor valor para C_{max} o $LB1$ de acuerdo con la variante escogida del algoritmo para ordenar la lista. Además, se eliminan los vértices cuya cota inferior sea igual o superior a una solución conocida (Z_0), se rechazan los vértices generados dominados por aquellos vértices ya escogidos previamente y se desplazan los vértices escogidos previamente dominados por los nuevos vértices. A través del procedimiento, el valor más pequeño de $LB1$ entre todas los vértices no escogidos se almacena en LBZ_{min}.

- Finalizar_etapa(): Consolida los vértices más prometedores de la etapa t (como máximo un número H de éstos).

Para ilustrar el funcionamiento del procedimiento *BDP* usaremos el ejemplo descrito en el apartado 3.3. Se ha aplicado un valor de $H = 6$ (el mismo número que piezas) y un límite superior $Z_0 = 40$ (una unidad por encima del valor de C_{max} de la secuencia óptima). Se ha usado la Variante 1. El resultado se puede encontrar en la Figura 3.4, donde además el procedimiento demuestra que la secuencia encontrada es óptima, ya que durante el proceso no se ha expulsado ningún vértice.

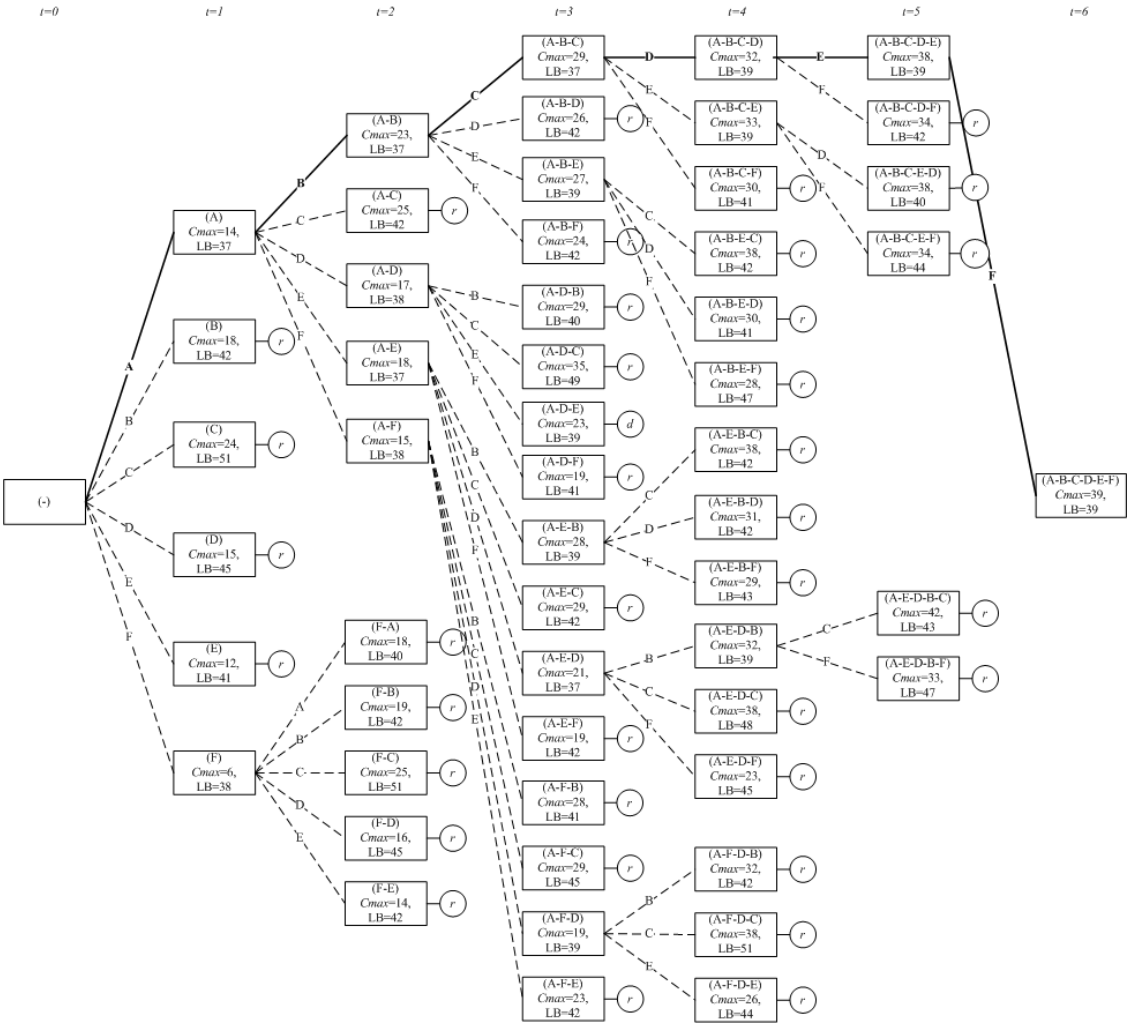

Figura 3.4: Grafo resultante del procedimiento *BDP* para el ejemplo, con $H = 6$, $Z_0 = 40$ y la Variante 1. "*d*" simboliza que el vértice está dominado y "*r*" que ha sido eliminado ($LB1 \geqslant Z_0$).

En el grafo de la Figura 3.4 podemos observar:

1. En esta ocasión no se desplaza ningún vértice. Esto ocasiona que sea posible asegurar que la secuencia $(A - B - C - D - E - F)$, con $C_{max} = 39$, es óptima.

2. Una gran parte de los vértices se eliminan debido a que su valor para la cota ($LB1$) es superior al valor introducido como Z_0. Algunos de los vértices que se eliminan son por ejemplo (D), ya que tiene un valor para $LB1 = 45$, o $(A-F-D-C)$, con $LB1 = 51$. Como se observa, se consigue reducir en gran medida los vértices, ya que gran parte de las soluciones menos prometedoras no se desarrollan.

3. El vértice que contiene la secuencia $(A - D - E)$ en la etapa $t = 3$ es dominado por la secuencia $(A - E - D)$ de la etapa $t = 3$, debido a:

- Las piezas secuenciadas son exactamente las mismas en ambos vértices.
- Instantes de finalización asociados a $(A-D-E)$: $e_{1,3}=14; e_{2,3}=19; e_{3,3}=23$.
- Instantes de finalización asociados a $(A-E-D)$: $e_{1,3}=14 : e_{2,3}=18; e_{3,3}=21$.

Los instantes de finalización asociados a $(A-E-D)$ son inferiores en todas las máquinas a los de $(A-D-E)$, lo que nos lleva, tal y como hemos comentado anteriormente, a que la secuencia $(A-E-D)$ domine a $(A-D-E)$, provocando que, como $(A-E-D)$ es una mejor combinación que $(A-D-E)$ (ya que deja más espacio libre para las piezas restantes), se elimine el vértice que contiene la secuencia $(A-D-E)$.

3.7. Experiencia computacional

Se ha realizado una experiencia computacional usando los 12 conjuntos de las instancias de benchmark de Taillard (Taillard, 1993). Estas instancias consisten en 120 instancias, agrupadas en 12 conjuntos. Cada uno de los conjuntos tiene 10 instancias, cada una de ellas con el mismo número de piezas y máquinas. El número de piezas va de 20 (conjunto 1) a 500 (conjunto 12), mientras que el número de máquinas va de 5 (conjunto 1) a 20 (conjunto 12).
Para obtener soluciones, se han usado las dos variantes propuestas en este capítulo del procedimiento *BDP*, programado en C++ y compilado usando gcc v.4.01, en un ordenador Apple iMac con un procesador Intel Core i7 2.93 GHz y 8 GB de RAM usando MAC OS X 10.6.4. Hay que destacar que ni el compilador ni la implementación utilizan ningún tipo de código paralelo, así que el ordenador puede ser considerado como un único procesador a 2.93 GHz. Los 12 conjuntos de instancias se han resuelto utilizando ocho anchos de ventana, H_1 a H_8 con valores 1, 10, 50, 100, 250, 500, 750 y 1000, respectivamente.
Como solución inicial Z_0 se ha usado la solución obtenida en el ancho de ventana anterior $H_{\alpha-1}$ para cada ancho de ventana H_α excepto en el caso con ancho $H_1=1$ en el que se fija Z_0 como ∞.
Para analizar los resultados, se ha usado la desviación porcentual relativa (RPD) calculada tal y como sigue:

$$RPD=\frac{BDP_{Best}-Best_{solution}}{Best_{solution}}\cdot 100 \tag{3.23}$$

La Tabla 3.2 muestra los resultados obtenidos, donde:

1. La columna "*Mejor lit*" muestra para cada una de las instancias (*Ins.* 1 a 120) de los conjuntos 1 a 12 de Taillard los mejores resultados recogidos en la literatura (ver (Ribas et al., 2011)).

2. La columna "*Mejor encon.*" muestra la mejor solución obtenida para cada Instancia usando las Variantes 1 y 2. En la mayoría de las instancias, la Variante 1 obtiene la mejor solución y únicamente en casos puntuales (marcados con el símbolo $^+$), la Variante 2 ofrece la mejor solución entre ellas.

3. Las columnas $H = 1$ a $H = 1000$ contienen la mejor solución para C_{max} obtenida para cada ancho de ventana.

4. Finalmente, la columna "*Mejor RPD*" muestra el mejor valor para RPD obtenido por *BDP* para cada una de las instancias.

	Ins.	*Mejor Lit.*	*H=1*	*H=10*	*H=50*	*H=100*	*H=250*	*H=500*	*H=750*	*H=1000*	*Mejor encon.*	*Mejor RPD*
			C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	C_{max}		
	1	1374	1640	1513	1441	1423	1390	1380	1380	1380	1380+	0.44
	2	1408	1725	1549	1474	1459	1450	1442	1432	1431	1431+	1.63
	3	1280	1667	1471	1353	1330	1326	1314	1302	1302	1302+	1.72
	4	1448	1563	1562	1543	1466	1456	1456	1456	1456	1456	0.55
Conjunto 1	*5*	1341	1512	1424	1400	1369	1367	1357	1350	1350	1350	0.67
n=20 m=5	*6*	1363	1515	1470	1395	1393	1393	1385	1385	1385	1385	1.61
	7	1381	1467	1407	1407	1396	1396	1396	1395	1393	1393	0.87
	8	1379	1608	1524	1392	1392	1386	1386	1386	1386	1386	0.51
	9	1373	1461	1432	1410	1410	1403	1403	1389	1389	1389	1.17
	10	1283	1421	1311	1307	1307	1293	1293	1293	1293	1293	0.78
	11	1698	2006	1807	1768	1762	1741	1741	1731	1731	1731	1.94
	12	1833	2116	1974	1974	1909	1909	1897	1895	1883	1883	2.73
	13	1659	1781	1728	1695	1687	1687	1684	1684	1683	1683	1.45
	14	1535	1791	1714	1640	1587	1587	1579	1579	1576	1576	2.67
Conjunto 2	*15*	1617	1978	1780	1738	1707	1707	1667	1667	1667	1667	3.09
n=20 m=10	*16*	1590	1830	1710	1611	1611	1610	1610	1610	1610	1610	1.26
	17	1622	1818	1740	1725	1722	1691	1691	1681	1681	1681	3.64
	18	1731	2133	1859	1796	1777	1761	1760	1752	1749	1749+	1.04
	19	1747	1962	1854	1854	1768	1755	1755	1755	1755	1755	0.46
	20	1782	2100	1933	1922	1890	1829	1829	1829	1829	1829	2.64

	21	2436	2772	2644	2640	2622	2567	2551	2551	2551	4.72
	22	2234	2760	2544	2429	2367	2350	2326	2315	2315	3.63
	23	2479	2813	2730	2705	2665	2663	2651	2644	2627	5.97
	24	2348	2733	2480	2440	2429	2419	2403	2388	2388	1.7
Conjunto 3	*25*	2435	2886	2740	2621	2602	2553	2534	2534	2534	4.07
n=20 m=20	*26*	2383	2744	2532	2492	2492	2492	2461	2461	2461	3.27
	27	2390	2956	2715	2672	2603	2603	2550	2518	2497	2497+	4.48
	28	2328	2792	2596	2574	2543	2522	2522	2522	2522	2522	8.33
	29	2363	3036	2570	2545	2494	2494	2483	2483	2483	2483	5.08
	30	2323	2698	2561	2442	2404	2404	2367	2367	2360	2360	1.59
	31	3002	3276	3146	3124	3096	3078	3066	3066	3066	3066	2.13
	32	3201	3481	3341	3267	3267	3253	3253	3253	3251	3251	1.56
	33	3011	3235	3146	3108	3081	3081	3081	3081	3077	3077	2.19
	34	3128	3554	3261	3261	3215	3187	3181	3181	3181	3181	1.69
Conjunto 4	*35*	3166	3471	3257	3226	3226	3226	3226	3216	3216	3216	1.58
n=50 m=5	*36*	3169	3530	3365	3360	3317	3317	3284	3284	3279	3279	3.47
	37	3013	3383	3188	3124	3098	3096	3096	3096	3090	3090	2.56
	38	3073	3480	3180	3125	3125	3125	3125	3125	3125	3125	1.69
	39	2908	3256	3107	3076	3005	3004	2986	2971	2971	2971	2.17
	40	3120	3434	3277	3217	3182	3171	3171	3163	3163	3163	1.38

	41	3638	4139	3911	3837	3744	3744	3730	3730	3730	2.53
	42	3507	3914	3701	3701	3647	3624	3624	3585	3571	1.82
	43	3488	3982	3848	3754	3754	3672	3642	3623	3623	3.87
	44	3656	4050	4024	3869	3869	3869	3869	3869	3840	5.03
Conjunto 5	*45*	3629	4109	3836	3761	3761	3761	3720	3712	3706	2.12
n =50 m=10	*46*	3621	3997	3845	3743	3743	3743	3692	3692	3692	1.96
	47	3696	4204	3940	3890	3814	3814	3814	3814	3814	3.19
	48	3572	4113	3986	3867	3718	3718	3718	3718	3709	3.84
	49	3532	3871	3742	3682	3660	3660	3636	3619	3619	2.46
	50	3624	4455	4011	3940	3883	3859	3811	3796	3780	4.3
	51	4500	5213	4899	4844	4753	4753	4741	4705	4705	4.56
	52	4276	5163	4960	4811	4720	4700	4700	4668	4658	8.93
	53	4289	5258	4879	4553	4553	4553	4553	4553	4553	6.16
	54	4377	5010	4663	4649	4643	4615	4572	4572	4572	4.46
Conjunto 6	*55*	4268	5291	4888	4669	4669	4624	4594	4542	4542	6.42
n =50 m=20	*56*	4280	5039	4876	4689	4651	4628	4596	4596	4594	7.34
	57	4308	5110	4853	4636	4636	4573	4505	4505	4505	4.57
	58	4326	5395	4836	4689	4627	4590	4544	4494	4494	3.88
	59	4316	5261	5044	4780	4780	4780	4732	4687	4680	8.43
	60	4428	5160	4887	4831	4719	4719	4663	4663	4639	4.77
	61	6151	6764	6417	6329	6270	6230	6225	6225	6225	1.2
	62	6022	6537	6236	6113	6113	6108	6108	6034	6034	0.2
	63	5927	6368	6207	5975	5975	5975	5942	5942	5928	0.02
	64	5772	6190	5926	5808	5805	5782	5782	5782	5755	***5755***	***-0.29***
Conjunto 7	65	5960	6453	6089	6070	6050	6050	6050	6016	5979	5979	0.32
n=100 m=5	66	5852	6471	6034	5945	5945	5876	5876	5876	5876	5876	0.41

	67	6004	6471	6220	6111	6081	6056	6056	6050	6046	6046	0.7
	68	5915	6397	6056	6002	5916	5916	5882	5882	5879	***5879***	***-0.61***
	69	6123	6647	6255	6255	6255	6201	6201	6172	6164	6164	0.67
	70	6159	6741	6274	6274	6244	6180	6154	6154	6154	***6154***	***-0.08***
Conjunto 8 *n=100* *m=10*	*71*	7042	7790	7404	7374	7283	7246	7231	7231	7103	7103	0.87
	72	6791	7547	7097	6957	6895	6866	6814	6814	6814	6814	0.34
	73	6936	7728	7293	7165	7157	7065	7050	7050	7050	7050	1.64
	74	7187	7925	7701	7553	7521	7482	7466	7405	7405	7405	3.03
	75	6810	7424	7110	7008	6962	6932	6932	6932	6932	6932	1.79
	76	6666	7427	7046	6971	6971	6934	6878	6855	6855	6855	2.84
	77	6801	7681	7322	7117	7117	7071	6983	6983	6983	6983	2.68
	78	6874	7415	7257	6998	6998	6998	6998	6972	6965	6965	1.32
	79	7055	7955	7453	7344	7281	7216	7216	7216	7216	7216	2.28
	80	6965	7705	7344	7225	7129	7129	7125	7123	7058	7058	1.34
Conjunto 9 *n=100* *m=20*	*81*	7844	9309	8982	8673	8560	8551	8479	8395	8395	8395	7.02
	82	7894	9234	8540	8380	8309	8248	8232	8232	8232	8232	4.28
	83	7794	9016	8664	8434	8434	8382	8334	8334	8303	8303	6.53
	84	7899	8891	8609	8604	8416	8244	8244	8240	8225	8225	4.13
	85	7901	9024	8378	8378	8225	8225	8203	8203	8185	8185	3.59
	86	7888	9241	8765	8553	8340	8340	8318	8318	8318	8318	5.45
	87	7930	8936	8620	8457	8457	8364	8364	8364	8241	8241	3.92
	88	8022	9386	8794	8681	8577	8534	8487	8449	8268	8268	3.07
	89	7969	8995	8626	8525	8357	8357	8205	8205	8205	8205	2.96
	90	7993	9275	8886	8771	8655	8655	8564	8432	8432	8432	5.49

	91	13406	14678	14089	13674	13674	13674	13557	13537	13468	13468	0.46
	92	13313	14472	13832	13592	13537	13311	13287	13287	13263	***13263***	***-0.38***
	93	13416	14463	13944	13727	13691	13615	13503	13475	13475	13475	0.44
	94	13344	14641	13981	13662	13644	13618	13550	13435	13435	13435	0.68
Conjunto 10	*95*	13360	14344	13962	13694	13566	13471	13426	13319	13319	***13319***	***-0.31***
n=200	*96*	13192	14115	13754	13484	13339	13304	13304	13273	13169	***13169***	***-0.17***
m=10	*97*	13598	15270	14161	13866	13805	13805	13728	13728	13675	13675	0.57
	98	13504	14831	13909	13782	13678	13629	13626	13626	13607	13607	0.76
	99	13310	14590	13735	13511	13468	13386	13386	13358	13298	***13298***	***-0.09***
	100	13439	14861	13914	13617	13600	13535	13523	13480	13456	13456	0.13
	101	14912	16617	16006	15558	15475	15331	15263	15263	15235	15235	2.17
	102	15002	16919	15860	15721	15718	15580	15501	15411	15351	15351	2.33
	103	15186	16484	15838	15568	15409	15324	15318	15318	15318	15318	0.87
	104	15082	16533	15762	15515	15506	15506	15430	15296	15296	15296	1.42
Conjunto 11	*105*	14970	17064	15823	15623	15569	15351	15351	15307	15307	15307	2.25
n=200	*106*	15101	16713	15968	15781	15781	15482	15482	15453	15453	15453	2.33
m=20	*107*	15099	16746	16075	15846	15840	15691	15567	15567	15522	15522	2.8
	108	15141	16430	15897	15594	15566	15534	15405	15388	15358	15358	1.43
	109	15034	16452	15896	15540	15540	15361	15361	15351	15351	15351	2.11
	110	15122	16385	15796	15522	15464	15400	15370	15370	15294	15294	1.14

	111	36609	38636	37081	36662	36576	36367	36366	36135	36135	***36135***	***-1.29***
	112	36927	39389	37636	37253	37067	36906	36906	36726	36703	***36703***	***-0.61***
	113	36646	38694	37426	36806	36582	36473	36408	36255	36174	***36174***	***-1.29***
	114	36641	38878	37508	36915	36856	36673	36506	36506	36416	***36416***	***-0.61***
Conjunto 12	*115*	36583	39016	37536	36916	36567	36567	36430	36289	36123	***36123***	***-1.26***
n=500	*116*	36917	39207	37959	37053	36985	36663	36618	36575	36501	***36501***	***-1.13***
m=20	*117*	36518	38606	37348	36658	36603	36395	36256	36256	36164	***36164***	***-0.97***
	118	36837	38727	37642	37064	36801	36561	36523	36503	36415	***36415***	***-1.15***
	119	36641	38761	37314	36791	36426	36426	36402	36235	36094	***36094***	***-1.49***
	120	36866	38866	37336	36772	36751	36485	36485	36418	36390	***36390***	***-1.29***

Tabla 3.2: Soluciones ofrecidas por el procedimiento *BDP* para cada uno de los anchos de ventana (H_1 a H_8)

Tabla 3.3: promedio para RPD y tiempos de CPU (en segundos) para los 12 conjuntos

Conjuntos	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
% promedio RPD 1	1.2	2.13	4.42	2.04	3.2	5.95	0.25	1.81	4.65	0.21	1.88	-1.11
% promedio RPD 2	2.03	4.3	5.8	3.25	5.24	9.59	3.42	4.22	8.66	4.31	6.61	6.2
% promedio RPD (ambos)	0.99	2.09	4.28	2.04	3.11	5.95	0.25	1.81	4.65	0.21	1.88	-1.11
promedio CPU 1/500	3.2	3.6	4.6	36.3	44.2	62.7	191.7	249	378.3	1625.2	2630.1	37046.8
promedio CPU 2/500	2.4	3.2	4.4	39.8	46.8	63.9	251.8	299.3	425.2	2156.9	2989.2	43280.7
promedio CPU 1/750	6.9	7.6	9.2	79.1	92.8	128.1	420.3	509.1	719.1	3190	4793.4	64481.6
promedio CPU 2/750	4.9	6.3	8.4	83.5	98.6	129.7	545	619.7	823.6	4415.5	5554.8	-
promedio CPU 1/1000	12.1	12.6	14.8	134.6	159.2	209.4	743.9	888.3	1147	5150.9	7389.7	97577.4
promedio CPU 2/1000	8	10.6	13.7	146.6	171.3	215.6	935.1	1061.8	1345.5	7219.8	8780.3	-

Tabla 3.4: Soluciones ofrecidas por el procedimiento *BDP* para los conjuntos 1 a 4 y anchos de ventana $H = 1250, 2500, 5000$ y 10000, para cada una de las instancias

	Ins.	*Mejor lit.*	C_{max} *H=1000*	*RPD H=1000*	*H=1250* C_{max}	*H=2500* C_{max}	*H=5000* C_{max}	*H=10000* C_{max}	*Mejor encon.*	*Mejor RPD*
Conjunto 1	*1*	1374	1380+	0.44	1379	1379	1379	1379	1379+	0.36
	2	1408	1431+	1.63	1431	1431	1431	1431	1431+	1.63
	3	1280	1302+	1.72	1302	1302	1290	1284	1284+	0.31
	4	1448	1456	0.55	1456	1456	1448	1448	1448	0
	5	1341	1350	0.67	1350	1349	1341	1341	1341	0
	6	1363	1385	1.61	1385	1371	1371	1371	1371	0.59
	7	1381	1393	0.87	1393	1393	1391	1386	1386	0.36
	8	1379	1386	0.51	1386	1386	1386	1382	1382	0.22
	9	1373	1389	1.17	1389	1378	1378	1373	1373	0
	10	1283	1293	0.78	1293	1283	1283	1283	1283	0
Conjunto 2	*11*	1698	1731	1.94	1730	1730	1730	1712	1712	0.82
	12	1833	1883	2.73	1879	1860	1852	1847	1847	0.76
	13	1659	1683	1.45	1683	1682	1682	1678	1678	1.15
	14	1535	1576	2.67	1572	1567	1557	1557	1557	1.43
	15	1617	1667	3.09	1667	1667	1664	1652	1652	2.16
	16	1590	1610	1.26	1610	1610	1606	1606	1606	1.01
	17	1622	1681	3.64	1654	1654	1651	1636	1636	0.86
	18	1731	1749+	1.04	1749	1749	1743	1740	1740+	0.52
	19	1747	1755	0.46	1755	1755	1755	1755	1755	0.46
	20	1782	1829	2.64	1829	1829	1817	1817	1817	1.96
Conjunto 3	*21*	2436	2551	4.72	2551	2537	2537	2537	2537	4.15
	22	2234	2315	3.63	2306	2284	2284	2270	2270	1.61
	23	2479	2627	5.97	2598	2598	2598	2592	2592	4.56
	24	2348	2388	1.7	2388	2388	2388	2386	2386	1.62
	25	2435	2534	4.07	2534	2515	2508	2486	2486	2.09
	26	2383	2461	3.27	2458	2444	2444	2438	2438	2.31
	27	2390	2497+	4.48	2497	2485	2479	2471	2471+	3.39
	28	2328	2522	8.33	2521	2512	2508	2502	2502	7.47
	29	2363	2483	5.08	2476	2458	2443	2432	2432	2.92
	30	2323	2360	1.59	2360	2360	2338	2338	2338	0.65
Conjunto 4	*31*	3002	3066	2.13	3066	3064	3063	3044	3044	1.4
	32	3201	3251	1.56	3251	3251	3251	3247	3247	1.44
	33	3011	3077	2.19	3077	3072	3047	3047	3047	1.2
	34	3128	3181	1.69	3181	3181	3181	3168	3168	1.28
	35	3166	3216	1.58	3216	3212	3205	3204	3204	1.2
	36	3169	3279	3.47	3269	3266	3254	3252	3252	2.62
	37	3013	3090	2.56	3090	3077	3077	3061	3061	1.59
	38	3073	3125	1.69	3125	3122	3098	3098	3098	0.81
	39	2908	2971	2.17	2971	2958	2958	2956	2956	1.65
	40	3120	3163	1.38	3163	3150	3150	3148	3148	0.9

Tabla 3.5: RPD promedio y tiempos de CPU (en segundos) para los conjuntos 1 a 4 y anchos de ventana $H = 1250, 2500, 5000$ y 10000

Conjunto	*1*	*2*	*3*	*4*
% promedio RPD	0.35	1.11	3.08	1.41
promedio CPU H=1250	17.17	19.92	23.56	231.62
promedio CPU H=2500	70.24	88.28	100.74	1094.28
promedio CPU H=5000	274.58	350.21	408.39	4484.2
promedio CPU H=10000	1058.99	1395.36	1590.45	19074.25

En la Tabla 3.2 se puede observar que los valores para la RPD se encuentran entre -1.49 % (instancia 119) y 8.93 % (instancia 52) para todas las instancias, considerando el mejor valor entre las dos variantes para el procedimiento *BDP*. Las valores negativos para la RPD (marcados en cursiva y negrita) indican una mejora a la mejor solución encontrada en la literatura. Estas mejoras ocurren en 17 instancias: instancia 70 (con un ancho de ventana $H = 500$), instancia 95 (con un ancho de ventana $H = 750$) y instancias 64, 68, 92, 96, 99, 111, 112, 113, 114, 115, 116, 117, 118, 119 y 120 (con un ancho de ventana $H = 1000$, aunque algunas soluciones obtenidas en anchos de ventana anteriores mejoraron también las mejores soluciones).

El promedio para la RPD de las 120 instancias es 2.18 %. La RPD promedio para cada uno de los conjuntos (1 a 12) y variante (1 y 2) se recoge en la Tabla 3.3. La Tabla 3.3 muestra también el promedio de tiempos de CPU (en segundos) para ambas variantes de la *BDP* y anchos de ventana $H = 500, H = 750$ y $H = 1000$ (para los 12 conjuntos). Los tiempos de CPU relacionados con $H \leqslant 250$ no se muestram debido a que son negligibles comparados con los tiempos de CPU expuestos.

La Tabla 3.3 nuestra que los tiempos de CPU aumentan si incrementamos el ancho de ventana y la dimensión de los conjuntos. En lo que respecta a los tiempos de CPU según la variante usada, en los conjuntos del 1 al 4 los tiempos son similares para ambas variantes. Sin embargo, desde el conjunto 5 al 12 la Variante 1 es más rápida.

Para estudiar el impacto del incremento del ancho de ventana H para mejorar las soluciones obtenidas, se ha realizado una experiencia computacional adicional usando los anchos de ventana $H = 1250, 2500, 5000$ y 10000 en las instancias de los conjuntos del 1 al 4 de Taillard. Los resultados de este experimento se recogen en las Tablas 3.4 y 3.5.

Las soluciones se han mejorado en 38 de las 40 instancias (excepto en las instancias 2 y 19), comparando los valores para la RPD de la columna "*Mejor RPD*" y "*RPD H=1000*", o alternativamente, comparando los valores para C_{max} de las columnas "C_{max} *H=1000*" y "*Mejor encon.*". Además, se ha alcanzado las soluciones óptimas de las instancias 4,5,9 y 10. Los tiempos de CPU para cada ancho de ventana ($H = 1250$ a 10000) se muestran en la Tabla 3.5. También se muestra "*% promedio RPD*" que muestra la mejora respecto a los valores obtenidos con $H = 1000$; estos valores son

0.64 %, 0.98 %, 1.2 % y 0.63 % para los conjuntos del 1 al 4, respectivamente.

3.8. Conclusiones del uso de la *BDP* para el problema *BFSP*

Para resolver el problema *BFSP* o $Fm|block|C_{max}$, que aparece en el contexto del problema del Taller mecánico cuando no existen *buffers* o espacios de almacenamiento entre máquinas, se ha propuesto un procedimiento basado en *BDP* (dos variantes, basados en la ordenación de las soluciones durante el procedimiento).
Para contrastar la calidad del procedimiento, se ha realizado una experiencia computacional con instancias de la literatura, para las cuales se conoce los mejores resultados obtenidos por los diferentes investigadores. En el caso del procedimiento *BDP*, ha mejorado las soluciones para 17 de las 120 instancias, en unos tiempos competitivos. Tras esta primera experiencia computacional, se mostró que la Variante 1 ofrecía, en casi todas las situaciones, mejores soluciones que la Variante 2.
Dado que el procedimiento parece prometedor, se ha realizado una experiencia computacional sobre las primeras 40 instancias de Taillard, para comprobar el efecto del incremento del ancho de ventana sobre la calidad de soluciones del procedimiento. En dicho experimento, podemos observar que se mejoran 38 de las 40 soluciones que se obtenían anteriormente con el procedimiento, alcanzando además 4 soluciones óptimas de la literatura.
Sin embargo, el procedimiento *BDP* todavía tiene posibilidad de mejora, como por ejemplo utilizando una mejor solución inicial como la propuesta en (Ribas et al., 2011) usando el método *MME*, o por ejemplo el uso de mejores cotas para el problema, como por ejemplo las cotas de Nabeshima, que aunque tienen un coste de computación más elevado (asumibles ya que los tiempos de computación del procedimiento son buenos), ofrecen cotas más ajustadas para el problema.

3.9. Comprobación de la calidad del procedimiento *BDP*: Algoritmos *GRASP* para el *BFSP*

Para observar el comportamiento de otros algoritmos en el problema, se ha implementado 7 procedimientos basados en *GRASP* similares a los descritos en el apartado 2.6. Estos procedimientos se basan en el esquema general presentado anteriormente (ver Algoritmo 6). En posteriores apartados se detallan las modificaciones realizadas sobre el procedimiento *GRASP*.

3.9.1. Fase *Greedy* de construcción de una solución

En este caso, la fase de construcción de soluciones es idéntica a la descrita en el apartado 2.6.2, pero empleando la siguiente cota como indicador de la calidad de la solución: Asumamos que se

ha construido una subsecuencia $\pi(t) = \{\pi_1, \pi_2, ..., \pi_t\}$ de t trabajos.
Supongamos ademas que disponemos de la información $x_i(t)$ y $e_k(t)$; donde $x_i(t)$ adopta el valor 1 si el trabajo i está presente en $\pi(t)$ y $e_k(t)$ representa el instante en que queda libre la máquina k tras procesar todos los trabajos contenidos en $\pi(t)$. El esquema de acotación empleado se muestra en la Figura 3.5.

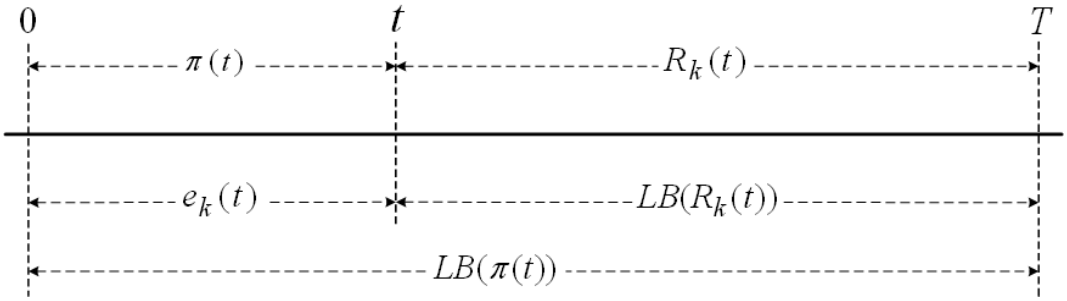

Figura 3.5: Esquema de acotación.

Para completar todos los trabajos, nos faltará añadir a la secuencia $\pi(t)$ un segmento conteniendo los $n - t$ trabajos pendientes no contenidos en $\pi(t)$. Si $R_k(t)$ es una secuencia de los trabajos pendientes en la máquina $k \in K$, es fácil obtener una cota inferior, $LB(R_k(t))$, para dicha secuencia:

$$LB(R_k(t)) = \sum_{i \notin \pi(t)} p_{i,k} + \underset{i \in \pi(t)}{min} \left\{ \sum_{\kappa=k+1}^{m} p_{i,\kappa} \right\} \tag{3.24}$$

Y, por consiguiente, una cota global de C_{max} asociada a la secuencia $\pi(t)$ será:

$$LB(\pi(t)) = \underset{k \in K}{max} \{e_k(t) + LB(R_k(t))\} \tag{3.25}$$

3.9.2. Fase de mejora local

Se propone una mejora local exhaustiva tipo 2-intercambio entre dos elementos de la secuencia en curso de mejora. La exploración del vecindario es determinista y se realiza de izquierda a derecha. A partir de una secuencia en curso, $\pi_c(T)$, con valor $C_{max}(\pi_c(T))$, se genera una secuencia vecina, $\pi_v(T)$, mediante un 2-intercambio tentativo entre los elementos que ocupan las posiciones t y t' de la secuencia en curso. Si $C_{max}(\pi_c(T)) > C_{max}(\pi_v(T))$, el intercambio se consolida, $\pi_v(T)$ se convierte en la nueva secuencia en curso y se reinicia el proceso de intercambios. En caso contrario, se prosigue con la generación de una nueva secuencia tentativa, una secuencia vecina, mediante otro 2-intercambio tentativo. El procedimiento finaliza cuando ninguna solución vecina tiene un valor de C_{max} mejor que el de la solución en curso.

3.9.3. Fase de mejora a través de *NEH*

Además, a la secuencia obtenida en la fase de mejora anterior, cuando se ha alcanzado un óptimo local, se puede aplicar el procedimiento *NEH* (Nawaz et al., 1983) consistente en seleccionar al azar un número fijo de elementos de la secuencia y extraerlos de la misma; posteriormente, los elementos extraídos se van insertando de uno en uno en la posición de la secuencia que genera menor C_{max}.

3.9.4. Experiencia computacional

Se ha realizado una experiencia computacional con las 6 primeros sets de ejemplares de Taillard (Taillard, 1993), para contrastar la aptitud de éstos algoritmos en el problema. Cada set contiene 10 ejemplares con el mismo número de trabajos (n) y mismo número de máquinas (m). En los 6 sets, el número de trabajos varía entre 20 (set 1) y 50 (set 6) y el número de máquinas entre 5 (set 1) y 20 (set 6). Como mejores soluciones se utilizan las soluciones de (Ribas et al., 2011), teniendo en cuenta las nuevas mejores soluciones obtenidas en el apartado 3.7.
Para obtener las soluciones de la fase constructiva se emplean 7 algoritmos derivados del procedimiento general *GRASP-x* para el que se han fijado los valores de los parámetros Z, L, f_0 y η (ver Tabla 3.6) siendo f_G el valor de referencia, para cada ejemplar, ofrecido por el procedimiento *greedy (G)* con tratamiento de empates. Los algoritmos resultantes tras asignar valores a los 4 parámetros son:

- (*G*) una heurística *Greedy* constructiva.
- (*M*) un procedimiento *Multistart* tradicional.
- (*GR-01/*$0{,}5 * |I|$) un algoritmo *GRASP* tradicional con una lista *RCL* limitada a $|I|/2$ candidatos (50 % del número de trabajos).
- (*GR-5/*$0{,}5*|I|$) un algoritmo *GRASP* con lista limitada a $|I|/2$ candidatos y con probabilidades de selección de éstos levemente dependientes de su aptitud.
- (*GR-8/*$0{,}5 * |I|$) un algoritmo *GRASP* con alta dependencia entre las probabilidades de selección de los candidatos y su aptitud y con lista restringida a $|I|/2$ candidatos.
- Dos algoritmos *Multistart*, (*GR-5/*$|I|$) y (*GR-8/*$|I|$) con probabilidades de selección de los candidatos idénticas a (*GR-5/*$0{,}5 * |I|$) y (*GR-8/*$0{,}5 * |I|$), respectivamente.

Cada uno de los 60 ejemplares se ha resuelto, primero, empleando las 7 variantes de la fase constructiva del algoritmo *GRASP* y, posteriormente, se han aplicado, a cada solución obtenida en la primera fase, dos procedimientos de mejora local. El primer procedimiento de mejora local (*LS* en el texto) es un 2-intercambio exhaustivo, consolidando el intercambio cuando se produce mejora.

Tabla 3.6: Algoritmos derivados de *GRASP-x*. Características.

Alg.	Z	L	f_0	η
G	1	1	∞	1
M	0.01	$\|I\|$	f_G/Z	1
GR-01/$0{,}5 * \|I\|$	0.01	$0{,}5 * \|I\|$	f_G/Z	1
GR-5/$0{,}5 * \|I\|$	0.5	$0{,}5 * \|I\|$	f_G/Z	1
GR-8/$0{,}5 * \|I\|$	0.8	$0{,}5 * \|I\|$	f_G/Z	1
GR-5/$\|I\|$	0.5	$\|I\|$	f_G/Z	1
GR-8/$\|I\|$	0.8	$\|I\|$	f_G/Z	1

El segundo procedimiento (*LS+NEH* en el texto) consiste en un bucle que aplica consecutivamente el primer procedimiento de mejora local mencionado, hasta alcanzar un óptimo local, y seguidamente la heurística *NEH* extrayendo e insertando 5 trabajos de la secuencia. El tiempo máximo de *CPU* concedido a la mejora de un ejemplar es de 10 s.

Los procedimientos *GRASP* han sido programados en *gcc v. 4.2.1*, en un ordenador Macintosh iMac con un procesador Intel Core i7, 2.93 Ghz. y 8 Gb de memoria RAM, usando MAC OS X 10.6.8 como sistema operativo. Ni la implementación ni el compilador hacen uso de *threads* ni de otra forma de código paralelo, y, por tanto, el ordenador actúa como un único procesador a 2.93 GHz.

En las Tablas 3.7 y 3.8 se muestran los resultados más significativos del experimento, empleando los procedimientos de mejora *LS* y *LS+NEH*, respectivamente, tras la fase constructiva del *GRASP*. En la Tabla 3.7 se recoge: (1) el número de óptimos alcanzado (*#opt*) por cada uno de los 7 algoritmos sobre los 60 ejemplares del *BFSP*; (2) el promedio de la *desviación porcentual relativa* (RPD=((*solución - óptimo*)/*óptimo*)$\times$100), para los ejemplares y cada algoritmo, tanto para la fase_1 *Greedy* constructiva del *GRASP* ($\overline{RPD_1}$) como para el proceso completo ($\overline{RPD_2}$) que incluye el procedimiento de mejora local (*LS* o *LS+NEH*); (3) el tiempo medio de *CPU*, por ejemplar, requerido por una iteración de cada uno de los 7 algoritmos *GRASP* ($\overline{CPU}$); y (4) el promedio para RPD para cada uno de los sets.

A la vista de las Tablas 3.7 y 3.8 observamos (en promedio para los 60 ejemplares) los siguientes hechos: (1) las soluciones obtenidas en la fase constructiva del *GRASP*, se alejan de las mejores soluciones conocidas aproximadamente entre el 13 % con el algoritmo *Greedy* determinista (G) y casi el 23 % con los algoritmos *Multistart* (M) y *GRASP* con $L = |I|$ ($GR - 5/|I|$ y $GR - 8/|I|$); (2) *LS* desciende, desde la solución ofrecida por la fase constructiva del algoritmo *Greedy* determinista (G), a óptimos locales con peor valor que los alcanzados por el resto de procedimientos que incorporan azar; (3) atendiendo a todos los procedimientos, *LS* mejora del 20.35 % al 3.23 % la

Tabla 3.7: $\#opt$, $\overline{RPD_1}$, $\overline{RPD_2}$, $\overline{CPU}$ y $\overline{RPD_2}$ para cada uno de los sets en el caso de *LS* para cada uno de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$	S1	S2	S3	S4	S5	S6
G	0	13.01	4.01	0.28	3.67	3.85	3.84	3.31	4.07	5.31
M	0	22.75	3.22	0.38	2.79	2.37	2.41	3.89	4.04	3.82
GR-01/$0{,}5 * \|I\|$	0	20.64	3.06	0.40	2.42	2.56	1.80	3.55	4.09	3.94
GR-5/$0{,}5 * \|I\|$	0	20.46	2.92	0.40	2.13	2.40	1.81	3.62	3.76	3.81
GR-8/$0{,}5 * \|I\|$	0	20.76	3.01	0.41	2.45	2.11	2.25	3.62	3.87	3.76
GR-5/$\|I\|$	0	22.43	3.26	0.37	2.58	2.43	2.00	4.17	4.26	4.10
GR-8/$\|I\|$	0	22.38	3.16	0.36	2.50	2.62	1.85	3.80	4.14	4.03

Tabla 3.8: $\#opt$, $\overline{RPD_1}$, $\overline{RPD_2}$, $\overline{CPU}$ y $\overline{RPD_2}$ para cada uno de los sets en el caso de *LS+NEH* para cada uno de los 7 algoritmos

	#opt	$\overline{RPD_1}$	$\overline{RPD_2}$	$\overline{CPU}$	S1	S2	S3	S4	S5	S6
G	19	13.01	0.69	10.00	0.07	0.12	0.16	0.95	1.27	1.56
M	25	22.75	0.70	10.00	0.13	0.00	0.03	1.10	1.37	1.57
GR-01/$0{,}5 * \|I\|$	20	20.64	0.64	10.00	0.14	0.13	0.07	0.85	1.25	1.43
GR-5/$0{,}5 * \|I\|$	24	20.46	0.68	10.00	0.10	0.11	0.00	1.04	1.29	1.53
GR-8/$0{,}5 * \|I\|$	28	20.76	0.66	10.00	0.00	0.00	0.02	1.12	1.32	1.53
GR-5/$\|I\|$	25	22.43	0.67	10.00	0.02	0.03	0.02	1.01	1.31	1.64
GR-8/$\|I\|$	25	22.38	0.71	10.00	0.04	0.12	0.02	1.20	1.35	1.52

distancia a las mejores soluciones conocidas; (4) *LS+NEH* reduce dichas distancias al 0.68 %; y (5) en todos los procedimientos con mejora *LS*, se alcanza el primer óptimo local en 0.37 s.

Además, en los procedimientos con mejora *LS*, no se alcanza ninguna mejor solución conocida para los 60 ejemplares. En cambio, usando todos los procedimientos con mejora local *LS+NEH* se alcanzan 29 sobre 60. Finalmente, el procedimiento $GR-8/0{,}5*|I|$ alcanza 28 óptimos, incluyendo los del set 1 y 2.

Éstos resultados prometedores nos llevaron a probar con el resto de instancias, pero debido al tamaño de los ejemplares, el procedimiento se estanca en la fase de búsqueda local para las instancias con más tamaño (debido a que en el tiempo concedido raramente se mejora la solución), con lo cual se ha llegado a la conclusión que es necesario sofisticar la búsqueda local para mejorar

los tiempos de computación.

3.10. Conclusiones

Los procedimientos *GRASP* implementados se muestran competitivos respecto a los existentes en la literatura. Concretamente el procedimiento $GR - 8/0{,}5 * |I|$ alcanza todas las mejores soluciones conocidas para los sets 1 y 2 de Taillard. El procedimiento *NEH* reduce las distancias al 0.68 % respecto a las mejores soluciones conocidas, mientras que su exclusión de los procedimientos producen una reducción al 3.23 %; sin optimización local los procedimientos constructivos dejan esta distancia al 20.35 %.

Sin embargo, como se ha comentado, para las instancias más grandes del problema el procedimiento se estanca en la fase de búsqueda local, debido al tamaño de las instancias, y normalmente no se mejora la solución. Esto nos indica que se ha de mejorar la fase de búsqueda local, con el fin de poder mejorar las soluciones en el tiempo concedido, con el fin de obtener mejores soluciones para estas instancias. Sin embargo, a pesar de este detalle, los algoritmos *GRASP* que incorporan la fase *NEH* se muestran competitivos en resultados comparados con el procedimiento *BDP*, con lo cual como trabajo futuro se tratará de incorporar algunas de las ideas de los algoritmos *GRASP* al procedimiento *BDP*, como por ejemplo la incorporación de una fase *NEH* a la hora de desarrollar los vértices.

No obstante, hay que tener el cuenta que el procedimiento *BDP* nos ha permitido mejorar 17 de las 120 instancias en un tiempo competitivo, con lo cual todavía creemos que podemos mejorar más el procedimiento, tanto en lo que se refiere a las cotas empleadas como a la incorporación de lo aprendido a través de los procedimientos *GRASP*.

Capítulo 4

El *ORV*

4.1. Introducción y estado del arte

Las líneas de montaje de productos mixtos son muy comunes en los contextos *JIT* y *DS* y permiten fabricar variantes de uno o más productos utilizando el mismo sistema productivo. Estas unidades de producto, a pesar de tener cierto grado de similitud (familias), pueden requerir diferente uso de recursos (recursos humanos, sistemas automatizados y herramientas, cómo es el caso del *MMSP-W*) y consumo de componentes en cada una de las estaciones de trabajo de la línea de montaje. Además, para obtener los productos finales, las partes que forman parte de cada uno de los productos (componentes), de acuerdo con la *BOM* (*bill of materials* o lista de materiales), se incorporan a la obra en curso siguiendo el curso de la línea.
Esta flexibilidad en la línea, debido a la variedad de productos, hace necesario determinar el orden en que las unidades de producto circulan por la línea, de acuerdo con tres principios generales:

1. Reducción del stock de componentes y productos semi-elaborados.
2. Uso eficiente del tiempo de fabricación disponible.
3. Reducción al máximo posible de la sobrecarga de trabajo.

Estos tres principios derivan en diferentes categorías de los problemas de secuenciación (Boysen et al., 2009):

1. Mixed-model sequencing problem.
2. Car sequencing problem.
3. Level scheduling problems.

Estas son las tres principales categorías de problemas de secuenciación. Nótese que esta tesis gira en torno a la resolución a través de algoritmos basados en *BDP* de alguno de los problemas

de estas categorías. En concreto, el problema sobre el que gira este capítulo es el problema *ORV* (Output Rate Variation, un problema real) (Monden, 1998), unido con propiedades deseables del problema *PRV* (Product Rate Variation, un problema académico) (Miltenburg, 1989).
En el problema *ORV* se han de secuenciar un número $|I|$ de tipos de producto. Cada tipo de producto tiene establecido una demanda d_i que debe ser producida. Además, hay un total de $|J|$ componentes usados en la fabricación de los diferentes productos (nótese que no es necesario que un producto use todos los componentes, por ejemplo en el caso de unidades con componentes especiales u opcionales). También se disponen como información la matriz N de elementos $n_{j,i}$, que nos indica la cantidad de componente j usada para fabricar una unidad de tipo i. Dados estos datos, es fácil calcular la tasa ideal de consumo para cada uno de los componentes.
El objetivo del problema *ORV* tradicional consiste en conseguir secuencias de unidades que minimizan la variación en la tasa de consumo de componentes usados por las unidades (entendiendo como variación la diferencia entre la tasa ideal de consumo y la tasa real para cada uno de los componentes). Esta función objetivo se puede medir de diferentes maneras, como por ejemplo la función *SDQ* (Bautista, 1993), (Bautista et al., 1996a) o *SDE* (Monden, 1998).
Una de las motivaciones para escoger este problema para esta tesis doctoral es que uno de los trabajos pioneros en el uso del procedimiento *BDP* (Bautista, 1993), resolvía el problema *ORV*. En esta tesis usaremos el procedimiento propuesto para solucionar el problema, debido a que en la literatura ya disponemos de ejemplares con soluciones de referencia (con lo cual no necesitaremos utilizar la programación lineal para encontrar un valor de referencia), añadiendo además las propiedades introducidas en el capítulo dedicado al *MMSP-W*, orientadas a favorecer la regularidad del mix de producción.
En contraposición, el objetivo del *PRV* consiste en obtener secuencias de unidades que regularicen la salida de productos de la línea de montaje. En algunas ocasiones, el *PRV* puede ser considerado como un caso particular del problema *ORV*.
Parte del trabajo presentado en este capítulo se puede encontrar en (Bautista, Alfaro and Cano, 2012; Bautista, Alfaro, Cano and Batalla, 2013).

4.1.1. Estado del arte

En (Boysen et al., 2009), encontramos una revisión de la literatura sobre el *ORV* enmarcada en el contexto de los problemas dentro del marco del *level scheduling*. A continuación se recogen algunos ejemplos de las heurísticas desarrolladas en la literatura para el problema *ORV*:

- (Monden, 1998) propuso en 1983 un algoritmo greedy llamado algoritmo *"Goal chasing"* para resolver el problema.
- (Bautista, 1993) y (Bautista et al., 1996a) propusieron un algoritmo basado en programación dinámica acotada (*BDP*).

- (Bautista, Companys and Corominas, 1996b) propusieron un método basado en el reparto proporcional.
- (Leu, Matheson and Rees, 1996) propusieron un algoritmo genético.
- (Balinski and Shahidi, 1998) propusieron un algoritmo basado en axiomas.
- (Jin and Wu, 2002) propusieron un algoritmo greedy y unas nuevas instancias de referencia basadas en las presentadas en (Bautista, 1993) y (Bautista et al., 1996a).
- (Miltenburg, 2007) propuso un algoritmo exacto para resolver el problema.
- (Bautista et al., 2010) compararon el comportamiento de un algoritmo basado en colonias de hormigas (*ACO*) con el procedimiento *BDP*.

Una línea de investigación muy trabajada ha sido la modificación del algoritmo *"Goal chasing"* propuesto por Monden con el objetivo de mejorar su rendimiento.

4.2. Formalización de los problemas *ORV* y *PRV*

4.2.1. El problema *ORV*

El problema *ORV* es descrito por primera vez en el trabajo (Monden, 1998) dedicado al sistema productivo de Toyota. El nombre *ORV* fue empleado por primera vez por (Kubiak, 1993). Para resolver este problema se han utilizado diversos procedimientos tal y como se ha mostrado en el apartado anterior.

El problema consiste en secuenciar, de forma regular, un total de D productos, agrupados en un conjunto I de tipos de producto, de los cuales d_i son del tipo $i (i = 1, ..., |I|)$. Además, los componentes se agrupan en un conjunto J. Una unidad de producto del tipo i requiere $n_{j,i}$ unidades del componente $j (j = 1, ..., |J|)$. El objetivo, tal y como se ha comentado, consiste en minimizar la variación en las tasas de consumo de todos los componentes durante la fabricación de los productos. Con ese fin, se puede definir la tasa ideal de consumo de componentes (constante en el tiempo) del componente j como (4.1) y el consumo ideal del componente j cuando se han fabricado t productos como (4.2):

$$\dot{n}_j = \frac{1}{D} \sum_{i=1}^{|I|} n_{j,i} \cdot d_i \qquad j = 1, ..., |J| \tag{4.1}$$

$$Y^*_{j,t} = \dot{n}_j \cdot t \qquad j = 1, ..., |J|; t = 1, ..., D \tag{4.2}$$

Además, cuando se han fabricado t productos, de los cuales $X_{i,t}$ son del tipo i $(i = 1, ..., |I|)$, el consumo actual del componente j $(j = 1, ..., |J|)$ se puede definir como:

$$Y_{j,t} = \sum_{i=1}^{|I|} n_{j,i} \cdot X_{i,t} \qquad j = 1, ..., |J|; t = 1, ..., D \tag{4.3}$$

La discrepancia o distancia entre el consumo ideal del componente j y el actual cuando han sido fabricadas en la línea t unidades de producto es:

$$\delta_{j,t}(Y) = Y_{j,t} - Y^*_{j,t} \qquad j = 1, ..., |J|; t = 1, ..., D \tag{4.4}$$

Podemos medir la no-regularidad del consumo de componentes para D productos usando las discrepancias definidas por la (4.4) de la siguiente manera:

$$\Delta_R(Y) = \sum_{t=1}^{D}\sum_{j=1}^{|J|}|\delta_{j,t}(Y)|, \Delta_E(Y) = \sum_{t=1}^{D}\sqrt{\sum_{j=1}^{|J|}\delta^2_{j,t}(Y)}, \Delta_Q(Y) = \sum_{t=1}^{D}\sum_{j=1}^{|J|}\delta^2_{j,t}(Y) \tag{4.5}$$

Donde $\Delta_R(Y)$, $\Delta_E(Y)$ y $\Delta_Q(Y)$, son las discrepancias globales en el consumo de componentes rectangular, euclídea y cuadrática, respectivamente.
Sea $\Im_Y = \{\Delta_R(Y), \Delta_E(Y), \Delta_Q(Y)\}$ el conjunto de funciones. Entonces, el modelo resultante mono objetivo para el *ORV* es:

$$Modelos \;\; M\_ORV: \qquad Min \quad f(f \in \Im_Y) \tag{4.6}$$

Sujeto a:

$$\sum_{t=1}^{D} x_{i,t} = d_i \qquad i = 1, ..., |I| \tag{4.7}$$

$$\sum_{i=1}^{|I|} x_{i,t} = 1 \qquad t = 1, ..., D \tag{4.8}$$

$$x_{i,t} \in \{0, 1\} \qquad i = 1, ..., |I|; t = 1, ..., D \tag{4.9}$$

En el caso de este modelo lineal, las restricciones (4.7) satisfacen la demanda de todos los productos; las restricciones (4.8) asignan sólo una unidad de producto a cada posición de la secuencia; y finalmente, las restricciones (4.9) fijan las variables $x_{i,t}(i = 1, ..., |I|; t = 1, ..., D)$ como binarias, tomando éstas el valor 1 si una unidad de producto del tipo i ocupa la *t-ésima* posición de la secuencia y 0 en cualquier otro caso. Obviamente, es posible establecer una relación entre las variables $x_{i,t}$ y $X_{i,t}$ de la siguiente manera: $X_{i,t} = \sum_{\tau=1}^{t} x_{i,\tau}$ $(\forall i = 1, ..., |I|; \forall t = 1, ..., D)$.

4.2.2. El problema *PRV*

El problema *PRV* se describe por primera vez en el trabajo de (Miltenburg, 1989) y su nombre se debe a (Kubiak, 1993). El problema se centra en secuenciar, de forma regular, un total de D

productos, agrupados en un conjunto I de tipos de producto, de los cuales d_i son del tipo i ($i = 1, ..., |I|$) de forma que las tasas de producción se mantengan lo más constante posible a lo largo del tiempo en los productos que se fabrican.
El *PRV* es un caso específico del *ORV* si se imponen las siguientes condiciones:

1. Una aplicación biyectiva entre los conjuntos I y J ($|I| = |J|$).
2. Cada tipo de producto requiere una unidad del componente relacionado a la aplicación anterior.

En dicho caso, se puede definir las siguientes funciones objetivo de no-regularidad en la producción (X) entre la producción ideal y la actual en el tiempo:

$$\Delta_R(X) = \sum_{t=1}^{D}\sum_{i=1}^{|I|}|\delta_{i,t}(X)|, \Delta_E(X) = \sum_{t=1}^{D}\sqrt{\sum_{i=1}^{|I|}\delta_{i,t}^2(X)}, \Delta_Q(X) = \sum_{t=1}^{D}\sum_{i=1}^{|I|}\delta_{i,t}^2(X) \tag{4.10}$$

donde:

$$\delta_{i,t}(X) = X_{i,t} - X_{i,t}^* = \sum_{\tau=1}^{t} x_{i,\tau} - \dot{d}_i \cdot t \qquad i = 1, ..., |I|; t = 1, ..., D \tag{4.11}$$

Siendo $\dot{d}_i = d_i/D$ ($i = 1, ..., |I|$) la tasa ideal de producción del tipo de producto $i \in I$ y $\delta_{i,t}(X)$ es la discrepancia o distancia entre la producción actual y la ideal del producto i cuando t unidades del producto han sido fabricadas.
Si se define $\Im_X = \{\Delta_R(X), \Delta_E(X), \Delta_Q(X)\}$ como el conjunto de funciones, el modelo resultante mono objetivo para el *PRV* es:

$$Modelos\ \ M\_PRV: \qquad Min\ \ f'(f' \in \Im_X) \tag{4.12}$$

Sujeto a:

$$(4.7) - (4.9)\ \ de\ \ M\_ORV \tag{4.13}$$

4.2.3. Relación entre los problemas *ORV* y *PRV*

El problema *PRV* es un caso particular del *ORV* cuando $I = J$ y $n_{j,i} = \delta_{j,i}$ (delta de *Kronecker*). Además, para establecer una unión entre las soluciones de ambos problemas, se usaran las propiedades derivadas de la preservación del mix de producción cuando se fabrican unidades de producto a lo largo del tiempo.
Sea $X_{i,t}^* = \dot{d}_i \cdot t$ el número de unidades del tipo de producto i ($\forall i \in I$), de un total de t ($\forall t \in D$) unidades que deberían ser producidas idealmente para mantener el mix de producción. Sea $\overrightarrow{X}^* = (X_{1,1}^*, ..., X_{|I|,D}^*)$ el punto ideal de la producción acumulada. Entonces, para el punto ideal $\overrightarrow{X}^*$ se cumple: $\delta_{i,t}(X) = X_{i,t} - X_{i,t}^* = 0$ ($\forall i, \forall t$) y además, $\Delta_R(X), \Delta_E(X)$, y $\Delta_Q(X)$ son óptimas y iguales a cero. Además, el punto $\overrightarrow{X}^*$ tiene la propiedad de regularizar el consumo de componentes:

Teorema 5. *Para el punto ideal* $\overrightarrow{X}^*$*:* $\delta_{j,t}(Y) = Y_{j,t} - Y^*_{j,t} = 0$ $(j = 1, ..., |J|; t = 1, ..., D)$.

Demostración. Tenemos: $Y_{j,t} = \sum_{i=1}^{|I|} n_{j,i} \cdot X^*_{i,t} \iff Y_{j,t} = \sum_{i=1}^{|I|} n_{j,i} \cdot \dot{d}_i \cdot t = t(\sum_{i=1}^{|I|} n_{j,i} \cdot \dot{d}_i)$ $= t \cdot \dot{n}_j = Y^*_{j,t}$. Entonces,

$$\delta_{j,t}(Y) = Y_{j,t} - Y^*_{j,t} = 0 \qquad j = 1, ..., |J|; t = 1, ..., D \tag{4.14}$$

□

Corolario 5. *Para el punto ideal* $\overrightarrow{X} = \overrightarrow{X}^*$ *se debe satisfacer* $\Delta_R(Y) = \Delta_E(Y) = \Delta_Q(Y) = 0$*. Consecuentemente, las funciones globales de discrepancia rectangular, euclidea y cuadrática del consumo de componentes son óptimas.*

4.3. Un ejemplo para el *ORV*

Para ilustrar el problema *ORV*, usaremos el siguiente ejemplo:
Existen dos tipos de productos ($|I| = 2$, $I = \{A, B\}$), cada uno de ellos con una demanda de 10 unidades ($d_A = 10$, $d_B = 10$, $D = d_A + d_B = 20$). Dichos productos utilizan tres tipos de componentes ($|J| = 3$, $J = \{C_1, C_2, C_3\}$). La lista de materiales para ambos productos se muestra en la Figura 4.1.

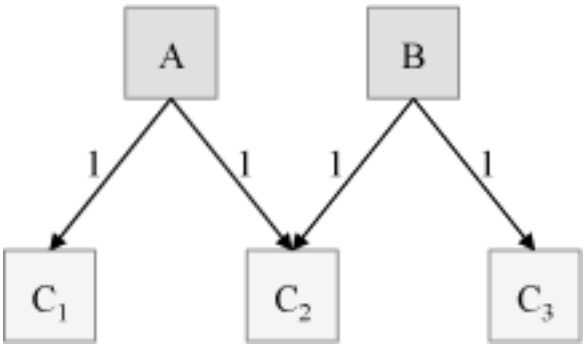

Figura 4.1: Lista de materiales (uso de los componentes) para los productos A y B.

La Tabla 4.1 muestra un resumen de los datos disponibles del ejemplo.

Si secuenciamos los 10 productos de tipo A y posteriormente los 10 productos de tipo B, el consumo resultante, así como el stock necesario para cada uno de los componentes se muestra en la Figura 4.2.

Tabla 4.1: Resumen de los datos del ejemplo. Lista de materiales y demanda para cada uno de los productos. Consumo y tasa ideal para cada uno de los componentes

Prod./Comp.	C_1	C_2	C_3	*Programa*
A	1	1	0	10
B	0	1	1	10
Consumo	10	20	10	20
$\dot{d}_i$	0.5	1	0.5	

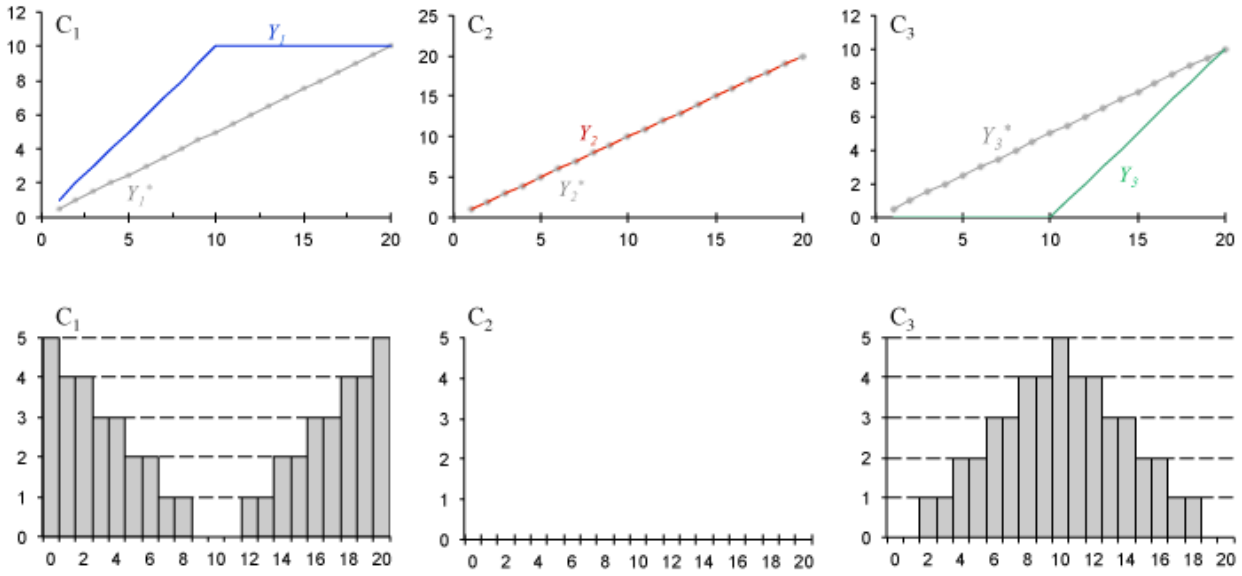

Figura 4.2: Parte superior: Consumo ideal (Y_j^*) y real (Y_j) para cada uno de los componentes para una secuencia 10 A + 10 B. Parte inferior: nivel de stock para cada uno de los componentes.

Como se puede observar en la Figura 4.2, si seguimos esta estrategia de secuenciación se producen diferencias entre las tasas ideales y reales para cada uno de los componentes, generando una gran necesidad de stock para los componentes C_1 y C_3.
En cambio, si secuenciamos los productos intercalando 5 productos A, 5 productos B, 5 productos A y finalmente 5 productos B, el consumo resultante, así como el stock necesario para cada uno de los componentes se muestra en la Figura 4.3.

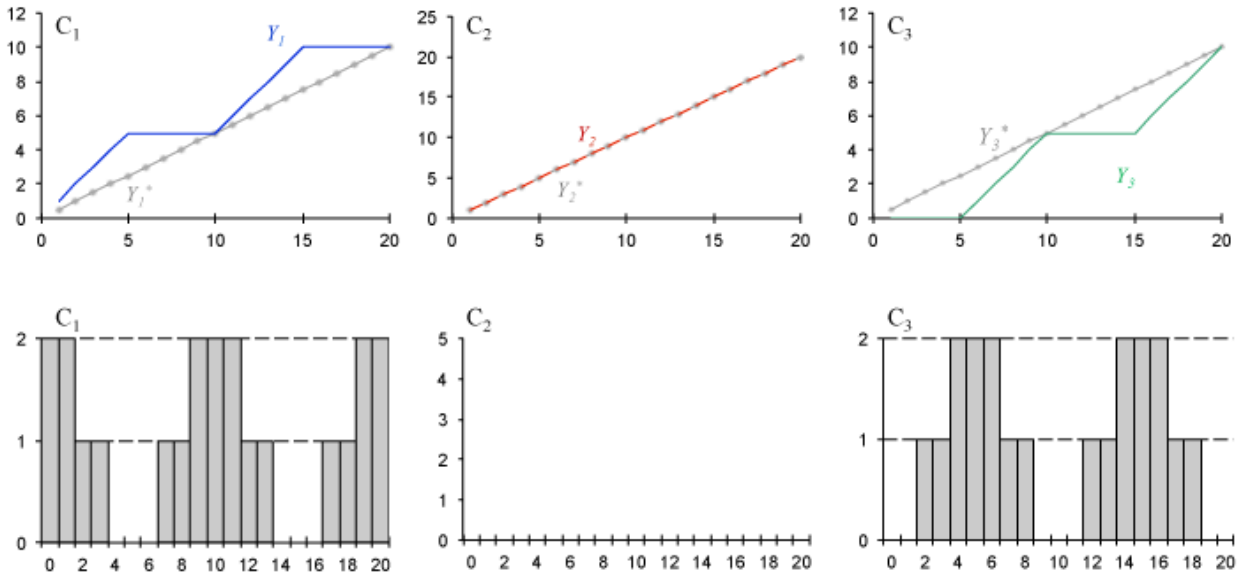

Figura 4.3: Parte superior: Consumo ideal (Y_j^*) y real (Y_j) para cada uno de los componentes para una secuencia 5 A + 5 B + 5 A + 5 B. Parte inferior: nivel de stock para cada uno de los componentes.

Como se puede observar en la Figura 4.3, si seguimos esta estrategia de secuenciación también producen diferencias entre las tasas ideales y reales para cada uno de los componentes, generando una necesidad de stock para los componentes C_1 y C_3, aunque en menor medida que usando la anterior estrategia de secuenciación.
Finalmente, si secuenciamos los productos intercalando un producto de tipo A y un producto de tipo B (secuencia $A-B-A-B-A-B-A-B-A-B-A-B-A-B-A-B-A-B-A-B$), el consumo para cada uno de los componentes se muestra en la Figura 4.4.

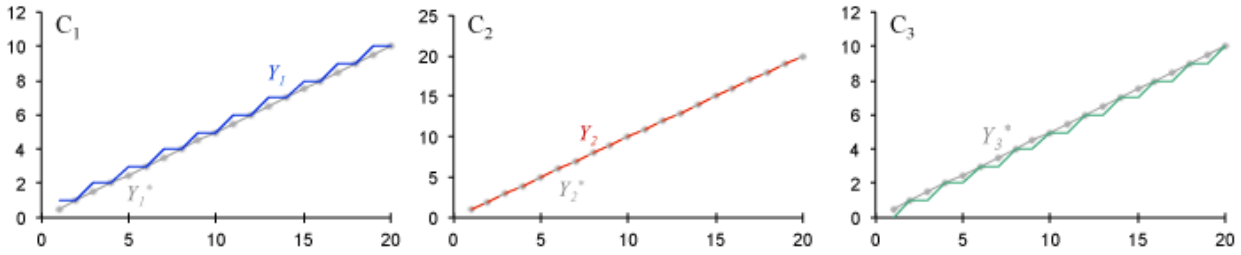

Figura 4.4: Consumo ideal (Y_j^*) y real (Y_j) para cada uno de los componentes para una secuencia 10 ($A + B$).

A la vista de la Figura 4.4, se puede observar que los consumos ideales son idénticos a los reales, generando que ante esta secuencia no sean necesarios stocks de ninguno de los componentes. Esta

secuencia corresponde con la solución óptima para el problema *ORV* del ejemplo.

4.4. Modelos para el problema *ORV* con regularidad en el mix de producción

Para resolver los problemas *ORV* y *PRV* a la vez, se pueden utilizar al menos dos vías de trabajo:

1. Resolver ambos problemas de forma conjunta como un problema multi-objetivo a través de la formulación y el empleo de nuevos modelos con funciones objetivo bi-objetivo.
2. Añadir a los modelos para el problema *ORV* original un conjunto de restricciones que garanticen la preservación del mix de producción.

4.4.1. Modelos bi-objetivo para el *ORV* y el *PRVP*

Basándonos en el Teorema 5 y las conclusiones que se pueden derivar de él, podemos afirmar que la preservación del mix de producción se alinea con la regularidad del consumo de componentes. Es por eso, que si ambas propiedades son deseables, es razonable formular los siguientes modelos bi-objetivo:

$$Modelos\ \ M\_ORV\_PRV:\ \ (Min\ \ f) \wedge (Min\ \ f') \qquad (f \in \Im_Y, f' \in \Im_X) \tag{4.15}$$

Sujeto a:

$$(4.7) - (4.9)\ \ de\ \ M\_ORV \tag{4.16}$$

4.4.2. Modelos para el *ORV* con restricciones en el mix de producción (*pmr*)

De nuevo a partir del Teorema 5, también podemos controlar la regularidad de la producción en las secuencias si limitamos los valores de las variables de producción acumulada, $X_{i,t}$ $(i = 1, ..., |I|$, $t = 1, ..., D)$, al valor entero más cercano a los valores ideales, $X^*_{i,t} = \dot{d}_i \cdot t$, debido a que estas variables deben ser enteras. Esto es:

$$\lfloor \dot{d}_i \cdot t \rfloor \leqslant X_{i,t} \leqslant \lceil \dot{d}_i \cdot t \rceil \qquad\qquad i = 1, ..., |I|; t = 1, ..., D \tag{4.17}$$

Donde $\lfloor x \rfloor$ y $\lceil x \rceil$ son el entero mas grande menor o igual que x y el entero mas pequeño mayor o igual que x, respectivamente.
Si se imponen las restricciones (4.17) a las secuencias, es posible derivar las siguientes propiedades:

Teorema 6. *Si* $\lfloor \dot{d}_i \cdot t \rfloor \leqslant X_{i,t} \leqslant \lceil \dot{d}_i \cdot t \rceil$, *(*$\forall i; \forall t$*), entonces* $X_{i,t} - X_{j,t} \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor$, $(\forall \{i, j\} \subseteq I; \forall t)$.

Demostración. Se satisface: $X_{i,t} \leqslant \lceil \dot{d}_i \cdot t \rceil$ y $\lfloor \dot{d}_j \cdot t \rfloor \leqslant X_{j,t}$, $(\forall \{i,j\} \subseteq I; \forall t)$.
Entonces, $X_{i,t} + \lfloor \dot{d}_j \cdot t \rfloor \leqslant \lceil \dot{d}_i \cdot t \rceil + X_{j,t} \iff X_{i,t} - X_{j,t} \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor$, $(\forall \{i,j\} \subseteq I; \forall t)$. □

Corolario 6. *Si $d_i < d_j$ entonces $X_{i,t} - X_{j,t} \leqslant 1$, $(\forall \{i,j\} \subseteq I; \forall t)$.*

Demostración. Usando el Teorema 6, tenemos $X_{i,t} - X_{j,t} \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor$. Además, $d_i < d_j \Rightarrow \lfloor \dot{d}_i \cdot t \rfloor \leqslant \lfloor \dot{d}_j \cdot t \rfloor$.
Por eso, podemos escribir: $X_{i,t} - X_{j,t} \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_i \cdot t \rfloor \leqslant 1$ $(\forall \{i,j\} \subseteq I; \forall t)$. □

Teorema 7. *Si $\lfloor \dot{d}_i \cdot t \rfloor \leqslant X_{i,t} \leqslant \lceil \dot{d}_i \cdot t \rceil$, $(\forall i; \forall t)$, entonces $X_{i,t} - X_{j,t} \geqslant \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil$, $(\forall \{i,j\} \subseteq I; \forall t)$.*

Demostración. Se debe satisfacer: $X_{i,t} \geqslant \lfloor \dot{d}_i \cdot t \rfloor$ y $\lceil \dot{d}_j \cdot t \rceil \geqslant X_{j,t}$, $(\forall \{i,j\} \subseteq I; \forall t)$
Además, $X_{i,t} + \lceil \dot{d}_j \cdot t \rceil \geqslant \lfloor \dot{d}_i \cdot t \rfloor + X_{j,t} \iff X_{i,t} - X_{j,t} \geqslant \lfloor d_i \cdot t \rfloor - \lceil d_j \cdot t \rceil$ $(\forall \{i,j\} \subseteq I; \forall t)$. □

Corolario 7. *Si $d_i > d_j$ entonces $X_{j,t} - X_{i,t} \leqslant 1$, $(\forall \{i,j\} \subseteq I; \forall t)$.*

Demostración. Usando el Teorema 7, tenemos $X_{i,t} - X_{j,t} \geqslant \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil$. Además, $d_i > d_j \Rightarrow \lceil \dot{d}_i \cdot t \rceil \geqslant \lceil \dot{d}_j \cdot t \rceil$.
Finalmente, podemos escribir: $X_{i,t} - X_{j,t} \geqslant \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil \geqslant \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_i \cdot t \rceil \geqslant -1 \Rightarrow X_{j,t} - X_{i,t} \leqslant 1$, $(\forall \{i,j\} \subseteq I; \forall t)$. □

Corolario 8. *Si $d_i = d_j$ entonces $|X_{i,t} - X_{j,t}| \leqslant 1$, $(\forall \{i,j\} \subseteq I; \forall t)$.*

Demostración. Usando el Teorema 6, $X_{i,t} - X_{j,t} \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor \leqslant \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_i \cdot t \rfloor \leqslant 1$.
Usando el Teorema 7, $X_{i,t} - X_{j,t} \geqslant \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil \geqslant \lfloor \dot{d}_j \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil \geqslant -1$.
Finalmente, $-1 \leqslant X_{i,t} - X_{j,t} \leqslant 1 \Rightarrow |X_{i,t} - X_{j,t}| \leqslant 1$. □

De esta forma, partiendo de los modelos de referencia *M_ORV*, tenemos:

$$Modelos \;\; M\_ORV\_pmr: \;\; Min \;\; f(f \in \Im_Y) \tag{4.18}$$

Sujeto a:

$$(4.7) - (4.9) \;\; de \;\; M\_ORV \;\; y \;\; (4.17) \tag{4.19}$$

En este apartado usaremos los modelos *M_ORV* y *M_ORV_pmr* con la función $\triangle_Q(Y)$ como función objetivo *f*.

4.5. El uso de la *BDP* para solucionar el *ORV* y el *ORV_pmr*

Para resolver el problema *ORV* y el *ORV_pmr* usaremos un procedimiento basado en programación dinámica (*BDP*). El procedimiento programado está basado en el procedimiento *BDP* propuesto por (Bautista, 1993) y (Bautista et al., 1996a) y incorpora diversos de los métodos para la resolución propuestos en dichos trabajos.
Para definir el grafo de estados, un vértice $J(t,j)$ de la etapa t contendrá la siguiente información:

- $\vec{x}(t,j) = (x_1(t,j), ..., x_n(t,j))$ es el vector de la demanda satisfecha de los trabajos secuenciados (o no) en el *j-ésimo* vértice de la etapa t, donde $x_i(t,j),\ \forall i \in I\ (i, ..., |I|)$ es el *i-ésimo* componente del vector $\vec{x}(t,j)$ que adopta un valor igual a la cantidad secuenciada de ese tipo de trabajo hasta la etapa t.

- $\triangle_Q(Y)(t,j)$ es la no-regularidad de todos los componentes acumulada en el vértice j de la etapa t.

El vértice $J(t,j)$ además satisface las siguientes propiedades:

$$\sum_{i=1}^{|I|} x_i(t,j) = t \tag{4.20}$$

$$0 \leqslant x_i(t,j) \leqslant d_i \qquad i = 1, ..., |I| \tag{4.21}$$

$$\lfloor \dot{d}_i \cdot t \rfloor \leqslant x_i(t,j) \leqslant \lceil \dot{d}_i \cdot t \rceil \qquad i = 1, ..., |I| \tag{4.22}$$

De nuevo, para reducir el tamaño del grafo de estados también se han utilizado reglas dominancia y equivalencia. Diremos que el vértice $J(t,j)$ domina o es equivalente a $J(t,j')$ si:

$$J(t,j) \prec J(t,j') \Leftrightarrow \left[\vec{x}(t,j) = \vec{x}(t,j')\right] \wedge \left[\triangle_Q(Y)(t,j) < \triangle_Q(Y)(t,j')\right] \tag{4.23}$$

$$J(t,j) \equiv J(t,j') \Leftrightarrow \left[\vec{x}(t,j) = \vec{x}(t,j')\right] \wedge \left[\triangle_Q(Y)(t,j) = \triangle_Q(Y)(t,j')\right] \tag{4.24}$$

En lo que respecta al esquema de acotación, se ha utilizado la cota definida en el trabajo de (Bautista et al., 1996a) conocida como $BOUND4$, aunque también se probaron previamente otras cotas, como la cota $BOUND5$ del mismo trabajo (basada en un algoritmo de transporte, y que fue descartada debido a que era mucho más lenta y no ofrecía mejores resultados sobre la empleada aquí).
Para el uso de la $BOUND4$, es necesario el uso de $q_{t,i}$, definida como la mínima contribución del producto i situado en la posición t de la secuencia (en el trabajo (Bautista et al., 1996a) se puede encontrar un procedimiento para calcular $q_{t,i}$, basado en el uso de la matriz de afinidad secuencial $A(t)$). Una vez obtenido los valores para $q_{t,i}$, estos se ordenan para una i dada de forma creciente, siendo $qo_{l,i}$ el valor que ocupa la posición l de dicha ordenación. La cota $BOUND4$ en la etapa t ($BOUND4(t)$) se puede calcular tal y como sigue:

$$BOUND4(t) = \frac{1}{2} \cdot \sum_{i=1}^{|I|} \sum_{l=1}^{d_i - X_{i,t}} qo_{l,i} - \frac{1}{2} \cdot A(t) \tag{4.25}$$

Donde $A(t)$ es la matriz de afinidad secuencial (también definida en (Bautista et al., 1996a)), que indica el incremento de $\triangle_Q(Y)$ del producto secuenciado en la etapa t, y $d_i - X_{i,t}$ representa la demanda pendiente del producto i.
Finalmente, para obtener una guía para el procedimiento *BDP*, podemos utilizar la siguiente expresión:

$$LBZ = \triangle_Q(Y_{t-1}) + A(t) + BOUND4(t) \tag{4.26}$$

donde $\triangle_Q(Y_{t-1})$ representa la no-regularidad hasta la etapa anterior, $A(t)$ la afinidad secuencial y finalmente $BOUND4(t)$ la cota calculada usando (4.25).
El procedimiento *BDP* utilizado se describe a continuación (ver más detalles en apartados anteriores):

Algoritmo 9 *Esquema BDP para el problema ORV_pmr*

Entrada: $D, |I|, |J|, d_i(\forall i), n_{j,i}(\forall i, \forall j), Z_0, H$
Salida: Lista de secuencias obtenidas con *BDP*

```
Inicialización: t = 0; LBZ_min = ∞
mientras t < T hacer
    t ← t + 1
    mientras lista de vértices consolidados en la etapa t − 1 (Λ(t − 1)) no vacía hacer
        Seleccionar_vértice(t)
        Desarrollar_vértice(t)
        Filtrar_vértices(Z_0, H, LBZ_min)
    fin mientras
    Finalizar_etapa()
fin mientras
```

Para resolver el *ORV_pmr* el procedimiento se ha adaptado desde el procedimiento anterior añadiendo un mecanismo para descartar, en cada etapa $(t = 0, ..., D)$, los vértices que no satisfacen las restricciones que garantizan la preservación del mix de producción (4.17). Esta regla de descarte reduce significativamente el tamaño del espacio de búsqueda de soluciones, debido a que el número de vértices $H(t)$ a considerar en cada etapa t del grafo está limitado por el número de tipos de producto $|I|$ de la siguiente manera:

$$H(0) = H(D) = 1; H(t) \leqslant \begin{pmatrix} |I| \\ \lceil |I|/2 \rceil \end{pmatrix} \qquad t = 1, ..., D-1 \tag{4.27}$$

Por ejemplo, en un conjunto de instancias con $|I| = 4$ para el ORV_pmr, sólo es necesario una ancho de ventana máximo de $H = 6$ para garantizar todos los óptimos.
Además, los Teoremas 6 y 7 (así como sus corolarios) nos permiten incorporar al procedimiento *BDP* diferentes bloques de reglas, varios de ellos con un efecto equivalente en el recorte del espacio de búsqueda. Definamos que cuando alcanzamos la etapa t, $X_j(\forall j)$ es la demanda satisfecha del vértice $J(t-1)$ que es seleccionado para ser desarrollado. Definamos $J(t,i) = J(t-1)\bigcup\{i\}$ como el nuevo vértice a explorar de la etapa t, construido añadiendo el tipo de producto i al vértice $J(t-1)$ que ha sido seleccionado para ser desarrollado. En estas condiciones, es posible definir el siguiente conjunto de bloques de reglas para descartar vértices:

- *BLOQUE 1:* Reglas provenientes de (4.17).
  Si $\exists j : (X_j < \lfloor \dot{d}_j \cdot t \rfloor) \vee (X_j > \lceil \dot{d}_j \cdot t \rceil) \rightarrow$ Descartar vértice

- *BLOQUE 2:* Reglas provenientes de los Teoremas 6 y 7.
  $\forall j \neq i$:Si $\exists j : X_i - X_j > \lceil \dot{d}_i \cdot t \rceil - \lfloor \dot{d}_j \cdot t \rfloor \rightarrow$ Descartar vértice
  $\forall j \neq i$: Si $\exists j : X_i - X_j < \lfloor \dot{d}_i \cdot t \rfloor - \lceil \dot{d}_j \cdot t \rceil \rightarrow$ Descartar vértice

- *BLOQUE 3:* Reglas provenientes de los Corolarios 6, 7 y 8.
  $\forall j \neq i$: Si $d_i < dj \wedge \exists j : X_i > X_j + 1 \rightarrow$ Descartar vértice
  $\forall j \neq i$: Si $d_i = dj \wedge \exists j : |X_i - X_j| > 1 \rightarrow$ Descartar vértice
  $\forall j \neq i$: Si $d_i > dj \wedge \exists j : X_j > X_i + 1 \rightarrow$ Descartar vértice

Para observar las características de la reducción del espacio de búsqueda, se usará el siguiente ejemplo: disponemos de tres productos A, B y C ($|I| = 3$). Las demandas para los tres productos son: $d_A = 3; d_B = 1; d_C = 2$. El grafo resultante para el ORV se puede observar en la Figura 4.5. En cambio, el grafo para el problema ORV_pmr al que se han introducido las restricciones en la preservación del mix de producción (4.17), el espacio de búsqueda se reduce como se indica en (4.27). El grafo resultante de esta reducción se puede observar en la Figura 4.6.

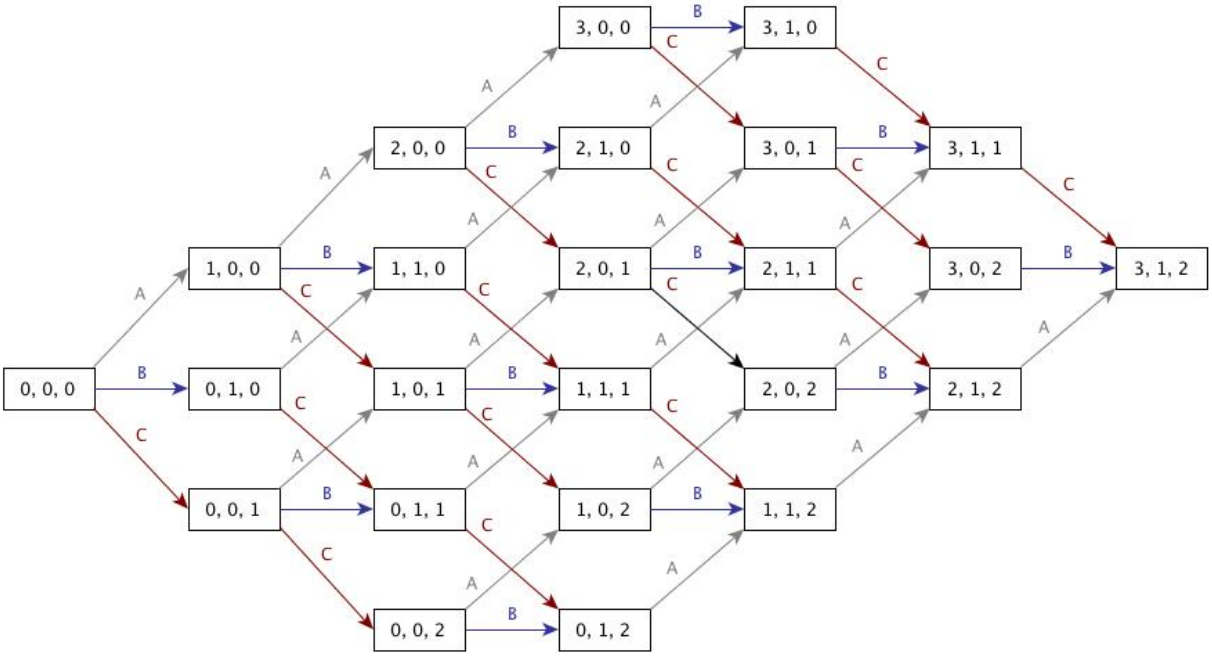

Figura 4.5: Grafo resultante para el ejemplo resolviendo el problema *ORV*

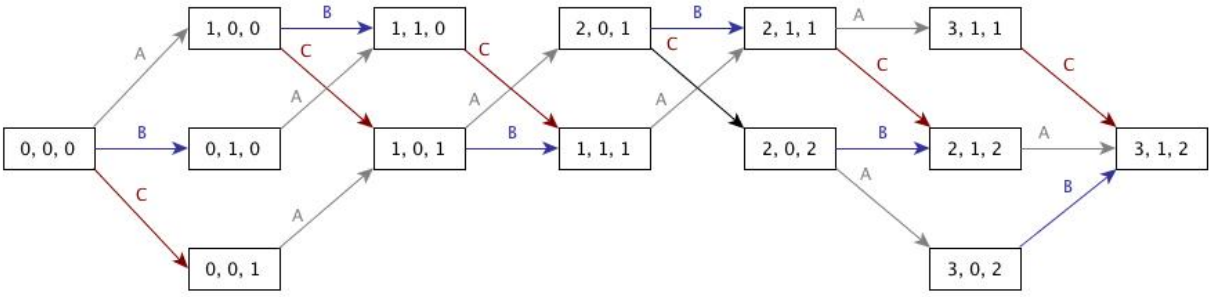

Figura 4.6: Grafo resultante para el ejemplo resolviendo el problema *ORV_pmr*, aplicando la reducción del espacio de búsqueda

Como se puede observar, se obtiene una drástica reducción del número de vértices pasando de los 24 vértices necesarios para el ORV a 14 en el caso del ORV_pmr.

4.6. Experiencia computacional

4.6.1. Experiencia computacional con instancias de referencia del *ORV*

La primera experiencia computacional corresponde a 225 instancias de referencia disponibles en (Jin and Wu, 2002). Estas instancias de referencia están basadas en las presentadas en (Bautista, 1993) y (Bautista et al., 1996a), multiplicando por 10 la demanda de cada uno de los productos. Dichas instancias disponen de 45 planes de producción agrupados en 5 bloques (B) y 5 estructuras de componentes de productos (E), que representan la lista de materiales. Todas las instancias tienen cuatro tipos de producto $(|I| = 4)$ y una demanda total de 200 unidades $(D = 200)$.
Para obtener las soluciones óptimas para los modelos *ORV* y *ORV_pmr*, se ha empleado el procedimiento *BDP* en las siguientes condiciones:

1. Procedimiento *BDP* programado en C++, usando gcc v4.2.1, ejecutado en un ordenador Apple iMac con un procesador Intel Core i7 a 2.93 GHz y 8 Gb de memoria RAM, usando MAC OS X 10.6.7, sin usar ningún tipo de código en paralelo.

2. Para alcanzar los óptimos se utilizaron 6 anchos de ventana $(H = 1, 6, 64, 128, 512, 1024)$, pero para demostrar todos se necesitó un ancho $H = 2048$ y $H = 4096$.

3. Como solución inicial, Z_0, para cada uno de los anchos de ventana, se utilizó la solución obtenida con *BDP* con el ancho de ventana anterior, excepto el caso $H = 1$, dónde se usó $Z_0 = \infty$.

Dados los modelos para el *ORV* y *ORV_pmr*, las funciones $\triangle_Q(Y)$, $\triangle_Q(X)$ y el conjunto de instancias E:

1. Se ha determinado la mejor solución para $\triangle_Q(Y)$ ofrecida para ambos modelos para cada una de las instancias $\epsilon \in E$, $S^*_{ORV}(\epsilon)$ y $S^*_{ORV\_pmr}(\epsilon)$;

2. Para estas mejores soluciones se ha obtenido la desviación porcentual relativa (RPD) para los valores de las funciones $f \in \{\triangle_Q(Y), \triangle_Q(X)\}$ tal y como se muestra en (4.28)

$$RPD(f, \epsilon) = \frac{f(S^*_{ORV}(\epsilon)) - f(S^*_{ORV\_pmr}(\epsilon))}{f(S^*_{ORV}(\epsilon))} \cdot 100 \quad (f \epsilon \{\triangle_Q(Y), \triangle_Q(X)\}; \epsilon \in E) \tag{4.28}$$

Los principales resultados del experimento se recogen en las Tablas 4.2, 4.3 y 4.4.
Como se puede observar en la Tabla 4.2, el algoritmo *BDP* que resuelve *ORV_pmr* es cincuenta veces más rápido que el que resuelve *ORV* en lo que respecta a tiempo de CPU promedio requerido para demostrar las soluciones óptimas. Además, el tiempo de CPU utilizado para solucionar el *ORV_pmr* no depende de la instancia resuelta.

Tabla 4.2: Tiempos de CPU mínimo, máximo y promedio necesarios para obtener las soluciones óptimas para los modelos *ORV* y *ORV_pmr* usando *BDP*.

	CPU_{min}	CPU_{max}	$\overline{CPU}$
ORV	0.64	80.03	11.32
ORV_{pmr}	0.17	0.22	0.21

Tabla 4.3: Número de óptimos alcanzados$^{(1)}$ y demostrados$^{(2)}$ para cada ancho de ventana (H).

H	1	6	64	128	512	1024	2048	4096
$ORV^{(1)}$	9	121	174	199	223	225	-	-
$ORV^{(1)}_{pmr}$	3	225	-	-	-	-	-	-
$ORV^{(2)}$	0	0	19	51	177	210	224	225
$ORV^{(2)}_{pmr}$	0	225	-	-	-	-	-	-

Tabla 4.4: Valores promedio de $RPD(\triangle_Q(Y))$ y $RPD(\triangle_Q(X))$.

	$E1$	$E2$	$E3$	$E4$	$E5$	*Promedio*
$RPD(\triangle_Q(Y))$	-5.02	-5.11	-1.60	-0.06	-10.92	-4.54
$RPD(\triangle_Q(X))$	39.24	19.03	5.11	0.17	37.03	20.11

En la Tabla 4.3 podemos observar los óptimos alcanzados y demostrados para los anchos de ventana utilizados. Para demostrar los óptimos de las instancias para el *ORV* fue necesario un ancho de ventana $H = 4096$, mientras que para alcanzarlos sólo necesitamos un ancho de ventana $H = 1024$. En cambio, para el *ORV_pmr* se alcanzaron y se demostraron todos los óptimos con un ancho de ventana $H = 6$.
Finalmente, en lo que concierne a la calidad de los resultados, la Tabla 4.4 nos muestra:

1. Un empeoramiento en promedio de 4.54 % para el valor óptimo de $\triangle_Q(Y)$ del *ORV_pmr* en comparación con el problema *ORV*.

2. La incorporación de las restricciones (4.17) (y los bloques de reglas asociados) mejoran en promedio un 20.11 % la preservación del mix de producción $(\triangle_Q(X))$.

3. Las ganancias en promedio en $\triangle_Q(X)$ y empeoramientos en promedio en $\triangle_Q(Y)$ más radicales se obtienen en aquellas estructuras de producto más alejadas en una equivalencia entre los problemas *ORV* y *PRV*.

4.6.2. Experiencia computacional con instancias del *CSP*

La segunda experiencia computacional corresponde con 9 instancias relacionadas al problema *Car Sequencing Problem (prob001)* disponibles en la librería *CSPlib* (www.csplib.org), sin tener en cuenta

las restricciones relacionadas con el *CSP*. En este caso, la demanda total es $D = 100$ unidades en todas las instancias y el número de componentes es de $|J| = 5$, mientras que el número de tipos de producto va de 22 a 26. Para estas instancias, el máximo tamaño de ancho de ventana requerido por *BDP* para demostrar los óptimos, cuando se incorporan las restricciones para preservar el mix de producción, oscilan entre $H \leqslant 705432$ (para $|I| = 22$) y $H \leqslant 10400600$ (para $|I| = 26$). Sin embargo, en esta experiencia computacional se ha empleado un ancho máximo de ventana $H = 1000$ tomando como soluciones iniciales (Z_0) las obtenidas con $H = 100$. Se han comparado los resultados obtenidos para el procedimiento *BDP* resolviendo los problemas *ORV* y *ORV_pmr*. Los resultados principales se encuentran en la Tabla 4.5:

Tabla 4.5: Valores para $(\triangle_Q(Y))$, $(\triangle_Q(X))$, $RPD(\triangle_Q(Y))$, $RPD(\triangle_Q(X))$ y cota inferior para $(\triangle_Q(Y))$ (LBZ_{min}) para los problemas *ORV* y *ORV_pmr*.

	ORV		*ORV_pmr*		LBZ_{min}	RPD	RPD
Instance	$(\triangle_Q(Y))$	$(\triangle_Q(X))$	$(\triangle_Q(Y))$	$(\triangle_Q(X))$	$(\triangle_Q(Y))$	$(\triangle_Q(Y))$	$(\triangle_Q(X))$
4/72	48.7	1107.6	51.1	374.5	43.1	-4.8	66.2
6/76	47.2	1403.1	48.9	313.0	42.1	-3.7	77.7
10/93	47.1	1095.3	49.4	399.3	41.7	-4.9	63.5
16/81	44.5	1374.6	47.6	442.2	41	-7.1	67.8
19/71	45.8	762.9	49.5	410.3	41.7	-8.1	46.2
21/90	46.8	1176.0	49.7	393.4	42.0	-6.2	66.5
26/82	47.3	1300.9	49.4	401.9	42.0	-4.4	69.1
36/92	45.0	1089.9	48.3	376.0	40.7	-7.3	65.5
41/66	45.3	1233.0	49.5	320.6	40.5	-9.2	74.0
Promedio						-6.2	66.3

Las soluciones obtenidas para los problemas *ORV* y *ORV_pmr* no son óptimas. Sin embargo, las cotas para $(\triangle_Q(Y))$ obtenidas para el *ORV_pmr* nos permiten encontrar soluciones para el *ORV* que se encuentran, en promedio para las 9 instancias, alrededor de un 10.24 % de la cota. La mejor solución obtenida corresponde a la instancia $16/81$, cuyo valor se encuentra a una distancia de un 7.87 % de su cota. La peor solución obtenida corresponde a la instancia $4/72$, con una distancia de 11.50 % de su cota.

Los tiempos de CPU promedios para cada una de las instancias es de 168*s*, para el *ORV*, y 96*s* para el *ORV_pmr*. Los tiempos de CPU para el *ORV_pmr* son mejores es más de un 40 % comparados con aquellos del *ORV*. Además, la mejora en la preservación del mix de producción $RPD(\triangle_Q(X))$ corresponde a un 66.3 %, mientras que la peora en el consumo regular de componentes $RPD(\triangle_Q(Y))$ corresponde a 6.2 %, cuando se incorporan al *ORV* las reglas relacionadas con la regularidad de la producción.

4.7. Conclusiones del uso de la *BDP* para los problemas *ORV* y *ORV_pmr*

Tal y como ya se conocía debido al trabajo de (Bautista et al., 1996a), los procedimientos *BDP* son competitivos para resolver el problema *ORV*. Para este capítulo, se ha implementado un nuevo procedimiento *BDP* basado en ese trabajo para demostrar la eficacia del procedimiento *BDP* para resolver una nueva variante del problema *ORV*, al que se le incorpora una propiedad muy deseada en las líneas de montaje mixtas: la regularidad en el mix de producción. A esta nueva variante se la denomina *ORV_pmr*.

Para resolver la nueva variante, se han propuesto nuevos modelos, teniendo en cuenta la resolución a través de funciones multi-objetivo y como nuevas restricciones del problema. Las propiedades que se han obtenido a través del estudio de la nueva variante se han agregado al procedimiento *BDP*.

El nuevo procedimiento *BDP* propuesto en este capítulo, en la primera experiencia computacional con 225 instancias de la literatura, demuestra que la incorporación de las restricciones en el mix de producción al problema original permiten reducir drásticamente los tiempos de CPU. Además, estas nuevas propiedades nos permiten acotar un ancho máximo de ventana para demostrar las soluciones óptimas para las instancias. En este caso, sabemos que en la nueva variante un ancho máximo de $H = 6$ es suficiente para demostrar todos los óptimos. Además, como es lógico, la introducción de las nuevas restricciones nos llevan a una peora en el consumo regular de componentes, ya que se ha de respetar la regularidad en el mix de producción. En el caso de esta experiencia computacional, el valor de la peora en consumo regular de componentes es de 4.54 %. Sin embargo, gracias a la introducción de las restricciones se ha logrado incrementar la regularidad en el mix de producción en un 20.11 %, así que las mejoras en regularidad de la producción son mucho más elevadas que las perdidas que podemos encontrar debido al empeoramiento del consumo regular de componentes.

En la segunda experiencia computacional, basada en 9 instancias de la librería *CSPlib* para el *CSP*, con una gran variedad de productos, nos ha permitido contrastar la calidad del nuevo procedimiento *BDP* cuando aumenta el número de productos. De nuevo, a pesar de que no se han podido alcanzar los óptimos (debido al tamaño de las instancias), se ha mejorado la regularidad del mix de producción en un 66.3 %, empeorando solamente un 6.2 % el consumo regular de componentes. De nuevo, el procedimiento *BDP* para solucionar el problema *ORV_pmr* se ha mostrado muy competitivo en comparación con el procedimiento para solucionar el problema *ORV* clásico.

Así que, como se ha podido observar en este capítulo, el procedimiento *BDP* para solucionar el problema *ORV_pmr* se ha mostrado muy competitivo, permitiendo encontrar en tiempos mucho menores que para el problema *ORV* clásico, soluciones con grandes mejoras en regularidad del mix de producción, empeorando sólo levemente la regularidad en el consumo de componentes. Además, las nuevas propiedades del problema permiten que el procedimiento encuentre en tiem-

pos muy pequeños (en el caso de la primera experiencia computacional) las soluciones óptimas para las instancias.

4.8. Otros procedimientos para el problema *ORV*

Para comparar la calidad del procedimiento *BDP* para resolver el problema *ORV* original, se programó un nuevo procedimiento *BDP* basado en el original presentado por (Bautista, 1993) y (Bautista et al., 1996a), utilizado como referencia en el apartado previo para obtener las soluciones para el *ORV*. Además, para compararlo con más procedimientos de la literatura se programaron los algoritmos *Goal Chasing*, propuesto por (Monden, 1998), y un algoritmo denominado *Variance*, propuesto por (Jin and Wu, 2002).
Para dar más variedad en la comparación de algoritmos, los profesores J. Pereira y J. Bautista programaron un algoritmo de hormigas (*MMAS-1*), que tal y como se ha comentado en el capítulo dedicado a la introducción, es un algoritmo de optimización que se basa en el comportamiento de las colonias de hormigas para obtener soluciones de calidad. Los detalles de dicho algoritmo de hormigas se puede encontrar en el trabajo (Bautista et al., 2010).
Para comparar la calidad de todos los procedimientos, se han utilizado las instancias propuestas por (Jin and Wu, 2002), utilizando 6 anchos de ventana para el algoritmo *BDP*. En las Tablas 4.6 y 4.7 se encuentra el resumen de la experiencia computacional realizada. En la Tabla 4.6 se puede encontrar el número de óptimos alcanzado y el tiempo de CPU necesario para cada uno de los procedimientos. La Tabla 4.7 muestra la comparativa del algoritmo *BDP* con el resto de procedimientos, indicando para cada una de las estructuras el valor medio de la función objetivo y el tiempo medio de CPU por ejemplar.

Tabla 4.6: Resultados globales del Experimento. Se muestra el número de óptimos y el tiempo de CPU (*s.*) por ejemplar

	Óptimos	*CPU*
Goal Chasing	0	-
Variance	0	-
BDP H=1	0	0.06
BDP H=100	35	4.91
BDP H=250	113	9,08
BDP H=500	176	12.15
BDP H=1000	210	14.11
BDP H=2000	224(*)	14.58
MMAS-1	101	16.35

Tabla 4.7: Cuadro resumen por estructuras del Experimento. Se muestra el valor medio de la función objetivo y el tiempo de CPU (*s.*) por ejemplar

	E1		E2		E3		E4		E5	
	Valor	CPU	*Valor*	CPU	*Valor*	CPU	*Valor*	CPU	*Valor*	CPU
Goal Chasing	893.05	-	4690.42	-	2666	-	171.71	-	12722.33	-
Variance	710.97	-	2266.2	-	1932.27	-	168.66	-	4637.36	-
BDP H=1	699.24	0.06	2266.94	0.06	1888.95	0.06	174.43	0.06	3346.08	0.06
BDP H=100	587.9	5.62	1298.52	5.26	1353.34	5.2	156.41	2.79	1450.64	5.68
BDP H=250	587.9	12.37	1298.52	8.78	1353.34	8.09	156.41	2.81	1447.34	13.35
BDP H=500	587.9	18.17	1298.52	9.29	1353.34	8.3	156.41	2.84	1447.15	22.14
BDP H=1000	587.9	20.42	1298.52	9.21	1353.34	8.23	156.41	2.88	1447.15	29.81
BDP H=2000	587.9	20.62	1298.52	9.21	1353.34	8.23	156.41	2.88	1447.15	31.96
MMAS-1	610.95	7.7	2213.21	22.14	1359.84	3.43	156.83	0.82	3020.2	47.67

A la vista de la Tabla 4.6 podemos observar que el procedimiento *BDP* se mostró más competitivo que los otros algoritmos, alcanzando y además demostrando las soluciones óptimas en 224 de las 225 instancias utilizando un ancho de ventana $H = 2000$ (con $H = 2500$, se garantizan los 225 óptimos). Además, se puede observar que los resultados ofrecidos por *MMAS-1* y *BDP H=250* (en número de óptimos), son similares, aunque el primer procedimiento empleó, en promedio por ejemplar, casi el doble de tiempo que *BDP*.

En lo que respecta a los resultados por estructuras, a la vista de la Tabla 4.7, el procedimiento *BDP* ofreció los mejores resultados en todas las estructuras en comparación con el resto de procedimientos, en unos tiempos competitivos. Además, se ha de tener en cuenta que el incremento de tiempo entre algunos anchos es debido a que el procedimiento *BDP* ya tenía la solución óptima, pero necesitaba un ancho de ventana mayor para demostrar el óptimo, característica que los otros procedimientos no pueden ofrecer ya que no son capaces de demostrar la optimalidad de la solución.

4.8.1. Conclusiones del uso de la *BDP* comparado con otros procedimientos

Como se ha podido observar, el procedimiento *BDP* se muestra muy competitivo para la resolución del problema *ORV* original, mejorando a dos heurísticas de la literatura y a procedimientos sofisticados como los algoritmos de hormigas, ya que ofrece mejores resultados que éstos en un tiempo competitivo, permitiendo además demostrar la optimalidad de las soluciones.

Es por esta calidad del procedimiento por lo que se escogió la *BDP* como procedimiento para resolver la variante del problema *ORV* a la que hemos llamado *ORV_pmr*.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones del trabajo

La presente tesis estudia el comportamiento de diversos procedimientos heurísticos y exactos para resolver diferentes problemas de optimización combinatoria.
Los problemas seleccionados corresponden al ámbito de la secuenciación de productos mixtos en líneas y talleres de montaje de productos en masa. Muchos de ellos corresponden al sector de la automoción por lo que se incorpora a esta tesis casos de estudio en plantas de fabricación de automóviles. Concretamente, estos problemas son:

1. *MMSP-W*, que corresponde a la secuenciación en línea de productos mixtos con el propósito de maximizar el trabajo completado cuando se considera la posibilidad de interrumpir las operaciones de ensamblado.

2. *ORV*, consistente en secuenciar unidades de producto con el propósito de reducir los stocks de los componentes en todos los eslabones de la cadena de suministro.

3. *BFSP*, consistente en secuenciar productos mixtos en módulos encadenados en línea sin almacenamiento intermedio (*buffers*) de la obra en curso con el objetivo de reducir al mínimo el tiempo de ocupación del taller o conjunto de módulos.

En cuanto a los procedimientos de resolución se han empleado básicamente tres tipos de algoritmos:

1. *BDP*. Procedimiento basado en la programación dinámica con empleo de cotas y reglas de dominancia entre soluciones para reducir el espacio de búsqueda.

2. *MILP*. Programación lineal entera mixta, procedimiento sobradamente conocido en la investigación operativa.

3. Procedimientos heurísticos en el marco del *Soft Computing*. Concretamente procedimientos *GRASP* con ponderación en el proceso de selección y un algoritmo *ACO*.

Los resultados cualitativos que se han obtenido quedan reflejados cualitativamente en la Tabla 5.1. En ella, para cada tipo de procedimiento descrito en la presente tesis se muestra el mejor o peor comportamiento (en una escala *A*, *B*, *C*) de cada tipo de procedimiento frente a cada tipo de problema. Dicho comportamiento tiene dos facetas: calidad de la solución y tiempo de computación. La calidad refleja la cercanía a la solución ideal mientras que el tiempo de proceso se vincula a la flexibilidad y rapidez del procedimiento ante el cambio de parámetros.

Tabla 5.1: Cuadro resumen del comportamiento cualitativo de los diferentes procedimientos para cada uno de los problemas. Escala (A, B, C) de mayor calidad a menor calidad.

	MMSP-W		*ORV*		*BFSP*	
	Calidad	*Tiempo*	*Calidad*	*Tiempo*	*Calidad*	*Tiempo*
BDP	*A*	*A*	*A*	*B*	*A*	*B*
MILP	*B*	Ej. Lit: *B* Ej. Nissan: *C*	-	-	-	-
Heurísticas	*C*	Ej. Lit: *A* Ej. Nissan: -	*B*	*A*	*A*	*A*

Sobre la Tabla 5.1 conviene hacer los siguientes comentarios:

- En lo que respecta el *MMSP-W*, el procedimiento que ofreció mejores resultados en calidad de soluciones para resolver las diferentes variantes del *MMSP-W* fue el procedimiento *BDP*, tanto para las instancias de la literatura como para las instancias correspondientes al caso de estudio de la planta de motores de Nissan, debido a que para los 225 ejemplares de la literatura encuentra los óptimos y obtiene buenas soluciones para las instancias correspondientes al caso de estudio. La programación lineal entera mixta ha obtenido una calificación de *B* debido a que aunque encuentra todos los óptimos para las instancias de la literatura, sólo encuentra soluciones para dos de las instancias de Nissan en un tiempo inferior a 7200 segundos. Finalmente, los algoritmos basados en *GRASP* fueron los que obtuvieron peor resultado. En lo que respecta al tiempo de computación, el procedimiento *BDP* y *GRASP* fueron los más rápidos. Sin embargo, en el caso de las instancias de la literatura los tiempos de computación eran pequeños en ambos casos, pero para las instancias de Nissan los algoritmos *GRASP* no eran lo suficiente rápidos. En lo que respecta a la programación lineal entera mixta, los tiempos de computación fueron más elevados, encontrando todos los óptimos de la literatura en tiempos razonables (aunque superiores a la *BDP*) y en el caso de las instancias de Nissan, hubo que limitar los tiempos de computación a 7200 segundos debido

al tamaño de éstas.

- En el problema *ORV*, la calidad de los resultados ofrecidos por el procedimiento *BDP* fue superior a la obtenida con los otros procedimientos de la literatura y el algoritmo basado en *ACO*. *BDP* tardó más tiempo en encontrar las soluciones que los procedimientos de la literatura, pero más rápida que el algoritmo *ACO*. No existen resultados de la programación lineal entera mixta.

- Por último, en el problema *BFSP* se han obtenido buenos resultados tanto con *BDP* como con los algoritmos *GRASP*. Para las instancias más pequeñas, *GRASP* obtuvo mejores resultados, pero sin embargo en las instancias más grandes *BDP* se comporto mejor. Esto es debido, entre otros factores, a la incorporación de *NEH* a la fase de búsqueda local de los algoritmos *GRASP*, pero la búsqueda local tarda demasiado para las instancias más grandes. De este dato podemos extraer que todavía queda campo de mejora en ambos procedimientos. Los tiempos de computación son ligeramente superiores en el procedimiento *BDP*, debido a que con este procedimiento también se resolvieron las instancias de mayor tamaño, mientras que los algoritmos *GRASP* sólo resolvieron las instancias más pequeñas debido a que era demasiado lento para las instancias de mayor tamaño. No existen resultados de la programación lineal entera mixta.

5.2. Trabajo futuro

Distinguimos entre dos vertientes:

5.2.1. Sobre el procedimiento *BDP*

En primer lugar, se explorarán nuevas propuestas de acotación de soluciones globales y parciales para las tres familias de problemas con el objetivo de reducir aún más el espacio de búsqueda, con la reducción consiguiente del tiempo de proceso y mejora de la calidad de las soluciones.
También se explorarán para las tres familias de problemas nuevas reglas de dominancia y pseudodominancia con los mismos propósitos que los descritos en el punto anterior.
Desde el punto de vista algorítmico una posible mejora para reducir los tiempos de computación es la paralelización en los cálculos aprovechando el esquema de expansión de la *BDP* y la capacidad de los procesadores actuales para ejecutar código en paralelo.
Además, se dotará a la *BDP* de capacidad de almacenamiento de vértices expulsados para posteriormente desarrollarlos si fuera necesario.
Finalmente, no debe descartarse la posibilidad de aplicar el procedimiento *BDP* a otras variantes de los problemas aquí tratados así como a otros problemas combinatorios.

5.2.2. Sobre otros procedimientos

Implementar otros procedimientos y mejorar los actuales, propios del *Soft Computing* con el objetivo de contrastar la calidad de las soluciones ofrecidas por éstos con la de la *BDP*.
Entre estos procedimientos están los diversos esquemas descritos en el apartado introductorio.

Anexos A

Publicaciones en el campo de estudio del autor de esta tesis

A.1. Artículos científicos

1. Bautista, J.; Cano, A.; Alfaro, R., 2012, Models for MMSP-W considering workstation dependencies: A case study of Nissan's Barcelona plant, European Journal of Operational Research, vol. 223, Elsevier, 669-679

2. Bautista, J.; Cano, A.; Companys, R.; Ribas, I., 2012, Solving the Fm/ block/ Cmax problem using Bounded Dynamic Programming, Engineering Applications of Artificial Intelligence, vol. 25/6, Elsevier, 1235–1245

3. Bautista, J.; Cano, A.; Alfaro, R., 2012, Modeling and solving a variant of the mixed-model sequencing problem with work overload minimisation and regularity constraints. An application in Nissan's Barcelona Plant, Expert Systems with Applications, vol. 39/12, elsevier, 11001–11010

4. Bautista, J.; Cano, A., 2011, Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules, European Journal of Operational Research, vol. 210/3, Elsevier, 495-513

A.2. Capítulos de libro

1. Bautista, J.; Cano, A.; Alfaro, R., 2014, Modeling and Solving a Variant of MMSP-W Problem with Production Mix Restrictions, Annals of Industrial Engineering 2012. Industrial Engineering: overcoming the crisis. Part IV. ISBN: 978-1-4471-5348-1, Springer, Holanda, pag. 235-243

2. Bautista, J.; Cano, A.; Alfaro, R.; Batalla, C., 2013, Impact of the Production Mix Preservation on the ORV Problem, Advances in Artificial Intelligence, Lecture Notes in Computer Science Volume 8109, Springer, Holanda, pag. 250-259

3. Bautista, J.; Alfaro, R.; Cano, A., 2013, Incorporating Regularity of Required Workload to the MMSP-W with Serial Workstations and Free Interruption of the Operations, Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services IFIP Advances in Information and Communication Technology Volume 397, ISBN: 978-3-642-40351-4 (Print) 978-3-642-40352-1 (Online), Springer, Holanda, pag. 405-412

4. Bautista, J.; Cano, A.; Alfaro, R., 2012, Performance of Bounded Dynamic Programming applied to a variant of the MMSP-W problem, Industrial Engineering: Innovative Networks (5th International Conference on Industrial Engineering and Industrial Management ÇIO 2011", Cartagena, Spain, September 2011, Proceedings), ISBN: 978-1-4471-2320-0, Springer, Holanda, pag. 339-348

5. Bautista, J.; Cano, A.; Companys, R.; Ribas, I., 2012, Solving flow shop problems with bounded dynamic programming, Industrial Engineering: Innovative Networks (5th International Conference on Industrial Engineering and Industrial Management ÇIO 2011", Cartagena, Spain, September 2011, Proceedings), ISBN: 978-1-4471-2320-0, Springer, Holanda, pag. 329-338

A.3. Congresos

1. Bautista, J.; Cano, A.; Alfaro, R.; Batalla, C., 2013, Impact of the Production Mix Preservation on the ORV Problem, Comunicación, Lecture Notes in Computer Science Volume 8109, pag. 250-259, XV Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA), Madrid

2. Bautista, J.; Cano, A.; Alfaro, R.; Batalla, C., 2013, Algoritmos GRASP para solucionar el problema Blocking Flow Shop, Comunicación, Actas. ISBN: 978-84-695-8348-7. Pág. 443-452, IX Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB 2013), Madrid

3. Bautista, J.; Alfaro, R.; Batalla, C.; Cano, A., 2013, Incorporating the Work Pace Concept into the MMSP-W, Comunicación, Industrial Engineering and Complexity Management. Book of Proceedings. ISBN: 978-84-616-5410-9, pags 427-435, VII International Conference on Industrial Engineering and Industrial Management, XVII Congreso de Ingeniería de Organización, Valladolid

4. Bautista, J.; Batalla, C.; Alfaro, R.; Cano, A., 2013, Impact of Ergonomic Risk Reduction in the TSALBP-1, Comunicación, Industrial Engineering and Complexity Management. Book of Proceedings. ISBN: 978-84-616-5410-9, pags 436-444, VII International Conference on Industrial Engineering and Industrial Management, XVII Congreso de Ingeniería de Organización, Valladolid

5. Bautista, J.; Cano, A.; Alfaro, R.; Batalla, C., 2013, Solving the Mixed Model Sequencing Problem with Workload Minimization with Product Mix Preservation, Comunicación, Industrial Engineering and Complexity Management. Book of Proceedings. ISBN: 978-84-616-5410-9, pags 409-417, VII International Conference on Industrial Engineering and Industrial Management, XVII Congreso de Ingeniería de Organización, Valladolid

6. Bautista, J.; Batalla, C.; Alfaro, R.; Cano, A., 2013, Extended Models for TSALBP with Ergonomic Risk Constraints, Comunicación, Preprints (CD), IFAC Conference on Manufacturing Modelling, Management, and Control (MIM 2013), San Petersburgo (Rusia)

7. Bautista, J.; Cano, A.; Alfaro, R.; Batalla, C., 2013, Hybrid procedure based on Bounded Dynamic Programming and Linear Programming for solving a variant of the MMSP-W, Comunicación, Preprints (CD), IFAC Conference on Manufacturing Modelling, Management, and Control (MIM 2013), San Petersburgo (Rusia)

8. Bautista, J.; Alfaro, R.; Cano, A., 2012, Incorporating Regularity of Required Workload to the MMSP-W with Serial Workstations and Free Interrumption of the Operations, Comunicación, Proceedings(CD), International Conference on Advances in Production Management Systems (APMS 2012), Rodas (Grecia)

9. Bautista, J.; Alfaro, R.; Cano, A., 2012, Solving the ORVP with Preservation of the Production Mix using BDP, Comunicación, Industrial Engineering: Overcoming the Crisis. Book of Full Papers. ISBN: 978-84-938642-5-5, pags 1554-1561, VI International Conference on Industrial Engineering and Industrial Management, XVI Congreso de Ingeniería de Organización, Vigo

10. Bautista, J.; Cano, A.; Alfaro, R., 2012, Modeling and Solving a Variant of MMSP-W Problem with Production Mix Restrictions, Comunicación, Industrial Engineering: Overcoming the Crisis. Book of Full Papers. ISBN: 978-84-938642-5-5, pags 1546-1553, VI International Conference on Industrial Engineering and Industrial Management, XVI Congreso de Ingeniería de Organización, Vigo

11. Bautista, J.; Cano, A.; Alfaro, R., 2012, Incorporando regularidad del trabajo requerido al MMSP con mínima sobrecarga, Comunicación, Industrial Engineering: Overcoming the Crisis. Book of Full Papers. ISBN: 978-84-938642-5-5, pag 1562-1569, VI International Conference

on Industrial Engineering and Industrial Management, XVI Congreso de Ingeniería de Organización, Vigo

12. Bautista, J.; Alfaro, R.; Cano, A., 2012, Modelos para el MMSP-W con estaciones en serie, procesadores paralelos, libre interrupción de operaciones y homogeneidad del trabajo requerido, Comunicación, Actas. Pág. 41, XXXIII Congreso Nacional de Estadística e Investigación Operativa, SEIO, Madrid

13. Bautista, J.; Cano, A.; Alfaro, R., 2012, Algoritmos GRASP para el MMSP-W con estaciones en serie y libre interrupción de operaciones, Comunicación, Actas. ISBN: 978-84-615-6931-1. Pág. 751-758, VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Albacete

14. Bautista, J.; Cano, A.; Alfaro, R., 2011, A bounded dynamic programming algorithm for the MMSP-W considering workstation dependencies and unrestricted interruption of the operations, Comunicación, Proceedings(CD). ISBN: 978-1-4577-1675-1, 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Córdoba

15. Bautista, J.; Cano, A.; Alfaro, R., 2011, Performance of Bounded Dynamic Programming applied to a variant of the MMSP-W problem, Comunicación, Proceedings. ISBN: 978-1-4471-2320-0, pag. 339-348, V International Conference on Industrial Engineering and Industrial Management, XV Congreso de Ingeniería de Organización, Cartagena

16. Bautista, J.; Cano, A.; Companys, R.; Ribas, I., 2011, Solving flow shop problems with bounded dynamic programming, Comunicación, Proceedings. ISBN: 978-1-4471-2320-0, pag. 329-338, V International Conference on Industrial Engineering and Industrial Management, XV Congreso de Ingeniería de Organización, Cartagena

17. Bautista, J.; Cano, A.; Companys, R.; Ribas, I., 2011, A Bounded Dynamic Programming algorithm for the Blocking Flow Shop problem, Comunicación, Actas del congreso (CD). pg. 8-15, ISBN: 978-1-61284-332-2., IEEE SSCI 2011 - IEEE Workshop on Computational Intelligence in Production and Logistics Systems (CIPLS), París

18. Bautista, J.; Pereira, J.; Cano, A., 2010, Algoritmos de hormigas para minimizar la variación de las tasas de fabricación en líneas de montaje de productos mixtos, Comunicación, pg. 253-260, ISBN: 978-84-92812-58-5, VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Valencia

Bibliografía

Aigbedo, H. and Monden, Y. (1997). A parametric procedure for multcriterion sequence scheduling for just-in-time mixed-model assembly lines, *International Journal of Production Research* **35**: 2543–2564.

Balinski, M. and Shahidi, N. (1998). A simple approach to the product rate variation problem via axiomatics, *Operations Research Letters* **22**: 129–135.

Bautista, J. (1993). Procedimientos heurísticos y exactos para la secuenciación en sistemas productivos de unidades homogéneas (contexto jit), *tesis doctoral UPC* .

Bautista, J., Alfaro, R. and Cano, A. (2012). Solving the orvp with preservation of the production mix using bdp, *Industrial Engineering: Overcoming the Crisis. Book of Full Papers. ISBN: 978-84-938642-5-5*, VI International Conference on Industrial Engineering and Industrial Management, XVI Congreso de Ingeniería de Organización, Vigo, Springer, pp. 1554–1561.

Bautista, J., Alfaro, R. and Cano, A. (2013). Incorporating regularity of required workload to the mmsp-w with serial workstations and free interruption of the operations, *Advances in Production Management Systems. Competitive Manufacturing for Innovative Products and Services IFIP Advances in Information and Communication Technology Volume 397, ISBN: 978-3-642-40351-4*, Springer, pp. 405–412.

Bautista, J., Alfaro, R., Cano, A. and Batalla, C. (2013). Impact of the production mix preservation on the orv problem, *Advances in Artificial Intelligence, Lecture Notes in Computer Science Volume 8109*, Springer, pp. 250–259.

Bautista, J. and Cano, A. (2011). Solving mixed model sequencing problem in assembly lines with serial workstations with work overload minimisation and interruption rules, *European Journal of Operational Research* **210**(3): 495–513.

Bautista, J., Cano, A. and Alfaro, R. (2012a). Algoritmos grasp para el mmsp-w con estaciones en serie y libre interrupción de operaciones, *Actas del congreso, ISBN: 978-84-615-6931-1*, VIII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Albacete, pp. 751–758.

Bautista, J., Cano, A. and Alfaro, R. (2012b). Modeling and solving a variant of mmsp-w problem with production mix restrictions, *Industrial Engineering: Overcoming the Crisis. Book of Abstracts*, VI International Conference on Industrial Engineering and Industrial Management, Springer, p. 231.

Bautista, J., Cano, A. and Alfaro, R. (2012c). Modeling and solving a variant of the mixed-model sequencing problem with work overload minimisation and regularity constraints. an application in nissan's barcelona plant, *Expert Systems with Applications* **39**(12): 11001–11010.

Bautista, J., Cano, A. and Alfaro, R. (2012d). Models for mmsp-w considering workstation dependencies: A case study of nissan's barcelona plant, *European Journal of Operational Research* **223**: 669–679.

Bautista, J., Cano, A., Alfaro, R. and Batalla, C. (2013). Algoritmos grasp para solucionar el problema blocking flow shop, *Actas del congreso, ISBN: 978-84-695-8348-7*, IX Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Madrid, pp. 443–452.

Bautista, J., Cano, A., Companys, R. and Ribas, I. (2012). Solving the fm/block/cmax problem using bounded dynamic programming, *Engineering Applications of Artificial Intelligence* **25**(6): 1235–1245.

Bautista, J. and Cano, J. (2008). Minimizing work overload in mixed-model assembly lines, *International Journal of Production Economics* **112**(1): 177–191.

Bautista, J., Companys, R. and Corominas, A. (1996a). Heuristics and exact algorithms for solving the monden problem, *European Journal of Operational Research* **88**(1): 101–113.

Bautista, J., Companys, R. and Corominas, A. (1996b). A note on the relation between the product rate variation (prv) problem and the apportionment problem, *The Journal of the Operational Research Society* **47**(11): 1410–1414.

Bautista, J., Companys, R. and Corominas, A. (1997). Modelling and solving the production rate variation problem (prvp), *TOP. Sociedad de Estadística e Investigación Operativa* **5**(2): 221–239.

Bautista, J. and Pereira, J. (2007). Ant algorithms for a time and space constrained assembly line balancing problem, *European Journal of Operational Research* **177**(3): 2016–2032.

Bautista, J. and Pereira, J. (2009). A dynamic programming based heuristic for the assembly line balancing problem, *European Journal of Operational Research* **194**(3): 787–794.

Bautista, J., Pereira, J. and Adenso-Díaz, B. (2008). A grasp approach for the extended car sequencing problem, *Journal of Scheduling* **11**(1): 3–16.

Bautista, J., Pereira, J. and Cano, A. (2010). Algoritmos de hormigas para minimizar la variación de las tasas de fabricación en líneas de montaje de productos mixtos, *Actas del congreso, ISBN: 978-84-92812-58-5*, VII Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, Valencia, pp. 253–260.

Bellman, R. (1954). The theory of dynamic programming, *Bulletin of the American Mathematical Society* **60**(6): 503–515.

Bellman, R., Esogbue, A. and Nabeshima, I. (1982). *Mathematical Aspects of Scheduling and Applications*, Pergamon Press.

Bolat, A. (2003). A mathematical model for sequencing mixed models with due dates, *International Journal of Production Research* **41**(5): 897–918.

Bolat, A. and Yano, C. A. (1992). Scheduling algorithms to minimize utility work at a single station on paced assembly line, *Production Planning and Control* **3**(4): 393–405.

Boysen, N. and Bock, S. (2011). Scheduling just-in-time part supply for mixed-model assembly lines, *European Journal of Operational Research* **211**(1): 15–25.

Boysen, N., Fliedner, M. and Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification and model critique, *European Journal of Operational Research* **192**(2): 349–373.

Cano, J., Ríos, R. and Bautista, J. (2010). A scatter search based hyper-heuristic for sequencing a mixed-model assembly line, *Journal of Heuristics* **16**(6): 749–770.

Caraffa, V., Ianes, S., Bagchi T, P. and Sriskandarajah, C. (2001). Minimizing makespan in a blocking flowshop using genetic algorithms, *International Journal of Production Economics* **70**(2): 101–115.

Carraway, R. L. and Schmidt, R. L. (1991). An improved discrete dynamic programming algorithm for allocating resources among interdependent projects, *Management Science* **37**(9): 1195–1200.

Companys, R. (2003). *Secuenciación*, CPDA, pp.99-100.

Companys, R. and Mateo, M. (2007). Different behaviour of a double branch-and-bound algorithm on fm—prmu—cmax and fm—block—cmax problems, *Computer and Operations Research* **34**(4): 938–953.

Ding, F., Zhu, J. and Sun, H. (2006). Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives, *International Journal of Production Economics* **102**(1): 108–131.

Erel, E., Gocgun, Y. and Sabuncuoglu, I. (2007). Mixed-model assembly line sequencing using beam search, *International Journal of Production Research* **45**(22): 5265–5284.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**: 109–133.

Festa, P. and Resende, M. (2009a). An annotated bibliography of grasp (part i: Algorithms), *International Transactions in Operational Research* **16**: 1–24.

Festa, P. and Resende, M. (2009b). An annotated bibliography of grasp (part ii: Applications), *International Transactions in Operational Research* **16**: 131–172.

Gilmore, P. C. and Gomory, R. E. (1964). Sequencing a one state-variable machine: A solvable case of the traveling salesman problem, *Operation Research* **12**: 655–679.

Gilmore, P. C., Lawler, E. L. and Shmoys, D. B. (1991). Well-solved special cases, *In E.L. Lawler; K.L. Lenstra; A.H.G. Rinooy Kan; D.B. Shmoys, The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* **Wiley**: 87–143.

Glover, F. and Kochenberger, G. (2003). Handbook of metaheuristics, *Eds. Kluwer Internacional Series* .

Gottlieb, J., Puchta, M. and Solnon, C. (2003). A study of greedy, local search and ant colony optimization approaches for car sequencing problems, *Lecture notes in computer science* **2611**: 246–257.

Grabowski, J. and Pempera, J. (2007). The permutation flow shop problem with blocking. a tabu search approach, *Omega* **35**(3): 302–311.

Graham, R. L., Lawler, E. L., Lenstra, J. K. and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey, *Annals of Discrete Mathematics* **5**: 287–326.

Hall, N. and Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no wait in process, *Opererations Research* **44**(3): 510–525.

Ibaraki, T. (1987). Enumerative approaches to combinatorial optimization, *Annals of Operations Research* **10-11**(1-4).

Jin, M. and Wu, S. (2002). A new heuristic method for mixed model assembly line balancing, *Computers and Industrial Engineering* **44**: 159–169.

Kotani, S., Ito, T. and Ohno, K. (2004). Sequencing problem for a mixed-model assembly line in the toyota production system, *International Journal of Production Research* **42**(23): 4955–4974.

Kubiak, W. (1993). Minimizing variation of production rates in just-in-time systems: A survey, *European Journal of Operational Research* **66**: 159–271.

Lageweg, B. J., Lenstra, J. K. and Rinnooy Kan, A. H. G. (1978). A general bounding scheme for the permutation flow-shop problem, *Operations Research* **26**(1): 53–67.

Laguna, M. and Martí, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization, *INFORMS Journal on Computing* **11**(1): 44–52.

Leisten, R. (1990). Flowshop sequencing problems with limited buffer storage, *International Journal of Production Research* **28**(11): 2085.

Leu, Y., Matheson, L. and Rees, L. (1996). Sequencing mixed model assembly line with genetic algorithm, *Computers and Industrial Engineering* **30**: 1027–1036.

Levner, E. (1969). Optimal planning of parts machining on a number of machines, *Automotive Remote Control* **12**(12): 1972–1978.

Liu, B., Wang, L. and Jin, Y. (2008). An effective hybrid pso-based algorithm for flow shop scheduling with limited buffers, *Computers and Operations Research* **35**(9): 2791–2806.

Marsten, R. E. . and Morin, T. L. (1978). A hybrid approach to discrete mathematical programming, *Mathematical Programming* **14**: 21–40.

McCormick, S. T., Pinedo, M. L., Shenker, S. and Wolf, B. (1989). Sequencing in an assembly line with blocking to minimize cycle time, *Operations Research* **37**: 925–936.

Miltenburg, J. (1989). Level schedules for mixed-model assembly lines in just-in-time production systems, *Management Science* **35**: 192–207.

Miltenburg, J. (2007). Level schedules for jit mixedmodel production lines: Characteristics of the largest instances that can be solved optimally, *Internacional Journal of Production Research* **45**: 3555–3577.

Monden, Y. (1998). *Toyota Production System, third ed*, Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, GA.

Morin, T. L. and Marsten, R. E. (1976). Branch-and-bound strategies for dynamic programming, *Operations Research* **24**(4): 611–627.

Nawaz, M., Enscore, J. E. and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem, *Omega* **11**(1): 91–95.

Okamura, K. and Yamashina, H. (1979). A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor, *International Journal of Production Research* **17**: 159–169.

Papadimitriou, C. H. and Kanellakis, P. C. (1980). Flowshop scheduling with limited temporary storage, *Journal of the ACM* **27**(3): 533–549.

Parello, B., Kabat, W. and Wos, L. (1986). Job-shop scheduling using automated reasoning, *Journal of Automated Reasoning* **2**(1): 1–42.

Prais, M. and Ribeiro, C. (2000). Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment, *INFORMS Journal on Computing* **12**: 164–176.

Putcha, M. and Gottlieb, J. (2002). Solving car sequencing problems by local optimization, *Lecture notes in computer science* **2279**: 132–142.

Qian, B., Wang, L., Huang, D. and Wang, X. (2009). An effective hybrid de-based algorithm for flow shop scheduling with limited buffers, *International Journal of Production Research* **47**(1): 1–24.

Rahimi-Vahed, A. and Mirzaei, A. H. (2007). A hybrid multi-objective shuffled frogleaping algorithms for a mixed-model assembly line sequencing problem, *Computers and Industrial Engineering* **53**(4): 642–666.

Reddi, S. S. and Ramamoorthy, B. (1972). On the flow-shop sequencing problem with no wait in process, *Operations Research Quarterly* **23**(3): 323–331.

Resende, M. and Ribeiro, C. (2003). *Greedy randomized adaptive search procedures, Handbook of Metaheuristics*, Kluwer Academic Publishers.

Ribas, I., Companys, R. and Tort-Martorell, X. (2011). An iterated greedy algorithm for the flowshop scheduling problem with blocking, *Omega* **39**: 293–301.

Ronconi, D. P. (2004). A note on constructive heuristics for the flowshop problem with blocking, *International Journal of Production Economics* **87**(1): 39–48.

Ronconi, D. P. (2005). A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking, *Annals of Operations Research* **138**(1): 53–65.

Scholl, A., Klein, R. and Domschke, W. (1998). Pattern based vocabulary building for effectively sequencing mixed-model assembly lines, *Journal of Heuristics* **4**: 359–381.

Taillard, E. (1993). Benchmarks for basic scheduling problems, *European Journal of Operational Research* **64**(2): 278–285.

Wang, L., Pan, Q., Suganthan, P. N., Wang, W. and Wang, Y. (2010). A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems, *Computers and Operations Research* **37**(3): 509–520.

Wang, L., Zhang, L. and Zheng, D. (2006). An effective hybrid genetic algorithm for flow shop scheduling with limited buffers, *Computers and Operations Research* **33**(10): 2960–2971.

Xiaobo, Z. and Ohno, K. (1997). Algorithms for sequencing mixed models on an assembly line in a jit production system, *Computers and Industrial Engineering* **32**: 47–56.

Yano, C. A. and Bolat, A. (1989). Survey, development, and application of algorithms for sequencing paced assembly lines, *Journal of Manufacturing and Operations Management* **2**(3): 172–198.

Yano, C. and Rachamadugu, R. (1991). Sequencing to minimize work overload in assembly lines with product options, *Management Science* **37**(5): 572–586.