TESIS DOCTORAL

PLANIFICACIÓN CONJUNTA DE LA PRODUCCIÓN Y DISTRIBUCIÓN DE PEDIDOS. MODELOS Y MÉTODOS DE RESOLUCIÓN

Autor: José Manuel García Sánchez
Ingeniero Informático por la E.T.S.I.I. de Sevilla

presentada en la ESCUELA SUPERIOR DE INGENIEROS de la UNIVERSIDAD DE SEVILLA

para la obtención del Grado de Doctor Ingeniero Industrial

Directores: Sebastián Lozano Segura
Fernando Guerrero López

Sevilla, Junio de 2003

A mis padres

ÍNDICE

AGRADECIMIENTOS

En primer lugar deseo expresar mi gratitud al Prof. D. Sebastián Lozano Segura, por su confianza, dedicación, exigencia, y por encima de todo, por su completa atención en todo momento durante el desarrollo de la tesis.

Mi agradecimiento igualmente al Prof. D. Fernando Guerrero López, por su confianza, ideas y ánimo constante durante toda la tesis.

Mi agradecimiento al Prof. D. Ignacio Eguia por su colaboración en materia de secuenciación. También les agradezco enormemente el apoyo y los consejos a todos mis compañeros del departamento que me han rodeado durante el desarrollo de mi trabajo.

Me gustaría también agradecer a la profesora Kate Smith por permitirme disfrutar de una estancia de 6 meses en la Universidad de Monash (Melbourne), donde desarrollé parte del trabajo de esta tesis.

Finalmente, quisiera mencionar muy especialmente a mi padres y a mis hermanos. Sin ellos, no hubiese sido posible realizar este trabajo.

RESUMEN

En esta Tesis se estudia el problema asociado a la planificación conjunta o coordinada de la producción y distribución de productos de carácter no almacenable. Estos productos se caracterizan por una distribución que debe realizarse de manera inmediata a la finalización del proceso productivo, debido a la ausencia de inventarios del producto. La fabricación se realiza en una planta y desde allí se distribuye el pedido hasta una localización determinada, con la ayuda de una flota de vehículos.

El problema se centra en la selección de los pedidos que van a ser servidos, de entre un conjunto de pedidos solicitados, debido a dos características fundamentales:

- El sistema funciona con una filosofía *just-in-time*, no permitiendo la entrega de un pedido fuera de su plazo de entrega, ya sea éste fijo o variable.
- Se imponen limitaciones de recursos, tanto en la fase de producción como en la de distribución, que normalmente impiden atender a todos los pedidos solicitados.

El análisis realizado sobre la actividad de los pedidos en el sistema encamina el problema hacia el campo de la secuenciación de trabajos en máquinas. De allí se extraen diversos modelos que ayudan al planteamiento y resolución del problema.

El problema se presenta sobre diferentes escenarios definidos en función del tipo de plazo de entrega (fijo o variable) y del número de plantas de producción. En cada escenario se presenta un modelo de programación entera, así como métodos exactos y aproximados para la resolución del problema. En todos los escenarios estudiados, los métodos aproximados permiten resolver el problema con errores mínimos y en tiempos aceptables, conclusión que nos resulta muy satisfactoria para los objetivos perseguidos en la investigación.

La presente memoria incluye un CD con todos los datos, programas informáticos y resultados.

Capítulo 1

Planteamiento de la investigación

Índice

1.1. INTRODUCCIÓN

La fabricación de productos de distribución inmediata es una actividad con una gran presencia en la sociedad de los últimos tiempos. Un ejemplo claro es el auge de actividades como el reparto de comida a domicilio y la distribución de productos industriales de aplicación inmediata, como puede ser el hormigón. En la literatura está muy desarrollado, aunque de forma independiente, el estudio de la secuenciación de trabajos en talleres de fabricación y el diseño de rutas de reparto. Sin embargo, el problema conjunto no ha sido lo suficientemente tratado.

La problemática asociada a los problemas de distribución de un determinado bien o servicio es muy variada dependiendo, entre otros factores, del carácter de la demanda, de la forma de atender las peticiones, de los plazos de entrega, y del número de almacenes desde donde se abastece la demanda.

La demanda de pedidos puede ser estable cuando el diseño de las rutas es fijo y establecido de antemano, o dinámica, en cuyo caso las peticiones llegan de forma impredecible sometiendo al sistema a continuos cambios. En general, se estudian problemas en los que la demanda es conocida y a partir de ella, se pretenden obtener las rutas de reparto que minimizan la distancia total recorrida por los vehículos, teniendo en cuenta las restricciones de capacidad de los mismos y permitiéndoles realizar varios itinerarios. También se estudian problemas en los que se prescinde de las restricciones de capacidad de los vehículos y se trata simplemente de minimizar la distancia recorrida. En estos problemas se presentan numerosas variantes, pero en todas ellas se considera admisible permitir suministrar varias peticiones en un mismo envío. Sin embargo, se pueden dar situaciones en las que las peticiones deban ser atendidas de forma independiente, debido a motivos de capacidad o a cuestiones operacionales. Esto provoca que el problema de distribución se aleje del diseño de rutas debiendo ser tratado como un problema de programación de vehículos más simplificado.

En cuanto a los plazos de entrega, principalmente se dan tres situaciones distintas según sea el compromiso temporal de suministro de los pedidos. En la primera situación no existe ningún tipo de restricción respecto a las fechas de entrega; por otro lado está el caso en el que existen límites o ventanas temporales para la entrega del pedido; como último caso está la situación en la que debe cumplirse necesariamente una fecha de entrega solicitada.

Un factor que condiciona fuertemente el problema de la distribución es el número de almacenes, plantas o depósitos desde donde se abastece la demanda. Cuando crece el número de éstos, la complejidad del problema aumenta, ya que además de

decidir el orden en el que se satisfacen las peticiones, también hay que decidir desde donde hacerlo.

Todo lo anterior pone de manifiesto que la problemática asociada al proceso de distribución de mercancías es muy variada, lo que hace que suela tratarse como un problema independiente del proceso de fabricación del producto que se distribuye.

Centrándonos en el proceso de fabricación del producto, los problemas se encuadran dentro del campo de la secuenciación de trabajos en máquinas. Se trata de establecer el programa de producción a corto plazo de las distintas instalaciones de la planta haciendo uso de un número limitado de recursos. En la bibliografía abundan las referencias sobre secuenciación de trabajos, pero en la mayoría de ellas no se impone como restricción la finalización de los trabajos en una fecha determinada, considerándose admisible cualquier instante del periodo de planificación.

En este proceso de estudiar independientemente los problemas asociados a la fabricación y a la distribución, los objetivos suelen ser de minimizar costes (de combustible, de tamaño de flota, laborales, etc.), maximizar la calidad del servicio (minimizar las desviaciones sobre la fecha de entrega solicitada) u otros objetivos que afectan a la gestión de la planta o de la flota de reparto (equilibrado de la producción o de la carga de trabajo de los vehículos). En estos objetivos los costes de transporte se consideran, generalmente, proporcionales a la distancia recorrida.

Abordar de forma independiente la producción de un bien y su distribución es únicamente posible cuando existen inventarios del producto fabricado. Cuando se da el caso de que el sistema productivo está acoplado al suministro debido a la ausencia de inventarios del producto final, hay que secuenciar la distribución en función de la capacidad de fabricación, de la disponibilidad de vehículos y de los datos de la demanda. Este caso surge cuando el producto ha de ser fabricado sobre pedido y servido inmediatamente, usualmente por su carácter perecedero o, en general, no almacenable. En estos problemas la distribución está supeditada al proceso productivo y, por ello, suele ser frecuente que este proceso sea bastante simplificado respecto a la obtención del producto final.

Normalmente, las peticiones pueden estar sometidas a restricciones temporales y en ocasiones tener carácter dinámico. La demanda suele estar muy concentrada en determinados momentos del día, lo que somete al sistema a una gran presión para responder de forma efectiva. Suelen existir varias plantas destinadas a la fabricación y desde las que se pueden servir los pedidos. Dada la corta vida del producto fabricado, el radio de acción de dichas plantas suele ser limitado, reduciéndose a un entorno urbano o metropolitano a lo sumo.

Ejemplos de estas situaciones se pueden encontrar en la producción y distribución de hormigón y en el reparto y preparación de comida rápida, ya que debido a la naturaleza de estos productos, los pedidos necesitan ser fabricados inmediatamente antes de ser repartidos al lugar de destino. El problema objeto de esta tesis se ocupa del estudio de esta clase de productos. En concreto, la problemática asociada a la producción y distribución de hormigón es la que se ajusta con mayor precisión a las características del problema que abordamos.

1.2. OBJETIVOS DE LA INVESTIGACIÓN

El objetivo principal de esta investigación es el estudio del problema asociado a la planificación coordinada de la producción y distribución de productos no almacenables. Debido a la ausencia de inventarios, estos productos se caracterizan por una fabricación y una distribución íntimamente ligadas. La fabricación se realiza sobre pedido y la distribución se produce inmediatamente después de la producción del pedido. El problema se ha denominado planificación conjunta de la producción y distribución de pedidos (PDP) y a la vista de la extensa revisión bibliográfica realizada, se puede afirmar que el problema no ha sido lo suficientemente estudiado en la literatura. Como contexto básico se ha usado la fabricación y distribución de hormigón *ready-mix*.

El objetivo básico del problema es determinar la selección de los pedidos que van a ser servidos, de entre un conjunto de pedidos solicitados. Dos características fundamentales impiden poder atender todas las peticiones:

1. Necesidad de respetar los plazos de entrega impuestos por el cliente

2. Recursos limitados en las fases de producción y distribución.

Una necesidad importante a la hora de acometer un problema con una perspectiva de estudio amplia, como es el caso del problema que nos ocupa, es la determinación tanto de los aspectos a tener en cuenta en el problema como de los aspectos que se descartan del análisis. Ello llevará a definir el problema con unas características determinadas que se asumen durante todo el trabajo. Sin embargo, también van a definirse determinados parámetros que influyen en la consideración del problema y en la presentación de diferentes escenarios de estudio. En concreto, los parámetros fundamentales analizados serán la rigidez en el plazo de entrega de los pedidos y el número de plantas de producción. A partir de estos parámetros se definen los siguientes escenarios:

- Escenario I: Problema PDP con instantes fijos de entrega y una planta de producción.
- Escenario II: Problema PDP con ventanas temporales de entrega y una planta de producción.
- Escenario III: Problema PDP con instantes fijos de entrega y varias plantas de producción.
- Escenario IV: Problema PDP con ventanas temporales de entrega y varias plantas de producción.

A su vez, el Escenario I del problema se dividirá en dos escenarios que difieren en el valor de otro de los parámetros analizados en el problema, el número de vehículos disponibles:

- Escenario IA: Se estudia el problema PDP con un número limitado de vehículos. Sería el caso general, en el que la selección de pedidos se ve limitada por la disponibilidad de vehículos y por la capacidad de producción de la planta.
- Escenario IB: Se plantea el problema PDP con un único recurso limitado, la capacidad de producción. El número de vehículos se considera ilimitado.

El análisis del problema PDP llevará a considerarlo como un problema dentro del amplio campo de la secuenciación de trabajos en máquinas, abstrayéndose, en función del escenario, a determinados modelos contemplados y no contemplados en la literatura. El estudio realizado sobre los modelos de secuenciación a los que se ajusta el problema va a servir como referencia para el planteamiento de determinados métodos de resolución.

Cada escenario se modelará mediante programación lineal entera y se resolverá de forma óptima y aproximada. Se usan diferentes herramientas de resolución que van desde el uso de la teoría de grafos para la resolución de problemas, hasta técnicas metaheurísticas y de exploración dirigida. El estudio sobre cada escenario se realiza de un modo progresivo atendiendo siempre a lo realizado en escenarios anteriores. Ello se pone sobre todo de manifiesto en el desarrollo de los escenarios III y IV (varias plantas), donde se asimilan con mayor grado las técnicas de resolución empleadas en escenarios precedentes.

En todos los escenarios y para todos los métodos desarrollados se realizarán múltiples experimentos computacionales con problemas de dificultad variada y generados aleatoriamente. El objetivo de tales experimentos es el de validar los enfoques heurísticos propuestos.

La generación de problemas se describe en el Anexo I de la tesis. Con el documento se incluye un CD con todos los experimentos computacionales. El contenido de dicho CD se describe en el Anexo II del documento.

Veamos a continuación el estudio específico realizado sobre cada escenario del problema:

- **Escenario IA**

La primera técnica que se implementa en este escenario es un método de resolución exacto basado en la construcción de un grafo cuyos caminos desde el nodo de inicio al nodo final determinan el conjunto de soluciones candidatas del problema. Este método servirá de base para los métodos exactos a definir en cada escenario del problema. El método se muestra ineficiente en consumo de memoria y tiempo con instancias grandes y cuando aumentan el número de recursos en el problema. Por ello se hace necesario la implementación de técnicas de carácter heurístico. Se implementan un algoritmo GRASP y una búsqueda tabú que amplia las características definidas en la búsqueda local del método GRASP. La búsqueda tabú corrige determinadas deficiencias que presenta el algoritmo GRASP sobre algunas instancias.

- **Escenario IB**

Este enfoque del problema es el que se muestra más “fácil” de resolver. Aquí no se necesitará la definición de algoritmos aproximados para obtener soluciones, puesto que un algoritmo de flujo a coste mínimo obtiene las soluciones óptimas del problema en un tiempo aceptable.

Sin embargo, dentro de este escenario se analizará también un caso particular más complejo que considera la minimización del uso de vehículos cuando se maximiza el número de pedidos servidos. Para este caso se define una heurística eficiente basada en un esquema de acotación y ramificación y, previamente, se implementa un método de solución exacta basado en un procedimiento iterativo de reducción de un grafo construido con la filosofía planteada en el Escenario IA.

- **Escenario II**

La existencia de ventanas temporales para la entrega de los pedidos aumenta la complejidad del problema. Ello se pone de manifiesto con la aplicación de un método de solución exacta. El método en este escenario estará basado también en la técnica utilizada en el Escenario IA. Además de este método, los procedimientos de carácter aproximado que se definen son:

- Dos enfoques basados en algoritmos genéticos.
- Un algoritmo de búsqueda tabú.

- **Escenario III**

Los escenarios con varias plantas de producción llevan el problema a una dimensión mayor de la contemplada en los escenarios anteriores. La dificultad estriba tanto en decidir qué pedidos procesar como desde qué planta hacerlo. En este escenario concreto se encuentran muchas analogías con el Escenario I y ello ayudará a su resolución.

Para este escenario se implementa, como en escenarios anteriores, un método de resolución exacto basado en la construcción de un grafo de estados admisibles de la planificación. Incluso para algunas instancias pequeñas, el método se muestra ineficiente. Por ello se implementa un procedimiento heurístico basado en las restricciones del modelo que emplea técnicas descritas en el Escenario I del problema.

Previamente se presentará un caso particular del problema que se resuelve de forma óptima en tiempo polinomial.

- **Escenario IV**

El escenario IV presenta el problema PDP en su forma más general (varias plantas y ventanas temporales). Las técnicas de resolución descritas aquí son fruto del trabajo realizado en los escenarios II y III del problema. Se desarrollará un método de solución exacta análogo al presentado en el Escenario III del problema y un algoritmo genético que reúne características de uno de los algoritmos descritos en el Escenario II, y del método heurístico presentado en el Escenario III.

Capítulo 2

Planificación conjunta de la producción y distribución de pedidos

Índice

2.1. FABRICACIÓN Y DISTRIBUCIÓN DE HORMIGÓN *READY-MIX*

La fabricación y distribución de hormigón *ready-mix* es una operación relativamente simple. Dependiendo del tipo de hormigón, una serie de materiales(cemento, arena, gravilla, agua y aditivos) se mezclan en diferentes proporciones y se cargan dentro del tambor montado de un vehículo, el cual inmediatamente distribuye el producto al lugar determinado por el cliente. Debido a que el producto es perecedero, existe un límite geográfico de cobertura de la planta. Por ello, es frecuente que una misma compañía posea diversas plantas apropiadamente distribuidas a lo largo del área metropolitana de servicio.

El proceso de producción es, por tanto, una operación basada simplemente en la mezcla de unos determinados materiales. Dicho proceso puede ser realizado previamente a la carga o incluso, que sea el propio vehículo el soporte para la realización de la mezcla, lo que induce al vehículo a formar parte del proceso productivo.

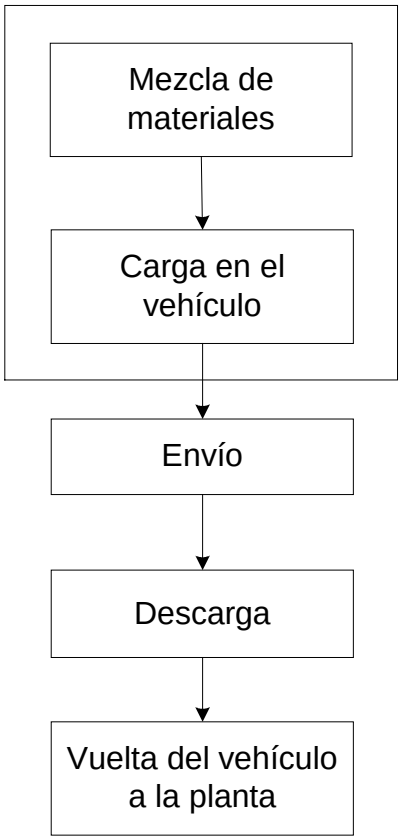


Figura 2.1: Actividad del proceso de fabricación y distribución de hormigón

El horizonte de planificación de la planta es un día. La demanda es usualmente conocida de antemano y corresponde al número de pedidos, cada uno de los cuales especifica el tipo y cantidad de hormigón a ser servido y la localización y momento en el que debe ser entregado.

Puesto que la planta tiene una capacidad de fabricación limitada, podría suceder que todas las peticiones no pudieran ser atendidas en el tiempo especificado por el cliente. Existen varias opciones disponibles entonces. Una es permitir cierta flexibilidad en las fechas de entrega (es decir, las fechas de entregas como intervalos de tiempo) de forma que pudiera ser servido el mayor número posible de pedidos. La otra es servir los pedidos desde diversas plantas situadas a lo largo de la zona geográfica de influencia de la empresa.

Diferencias con los sistemas de reparto de comida a domicilio

Las características de los sistemas de reparto de comida rápida son las siguientes:

- La producción de los pedidos se realiza generalmente desde diversos puntos, pero la asignación del punto de preparación para un pedido se realiza de antemano en función de la localización del mismo. Por ello, el problema puede ser presentado bajo el punto de vista de un solo punto de producción.

- Cada pedido lleva asociado una fecha de entrega calculada a partir del momento de realizar la petición y una ventana temporal dentro de la que se considera admisible la entrega. Sin embargo, generalmente se atienden la totalidad de pedidos que se reciben, aunque ello implique la entrega fuera de la ventana temporal. Ello implica asumir unos costes que suelen ser equivalentes al valor del pedido.

- Las peticiones llegan al sistema de forma impredecible, planteando una situación no determinista.

- La producción y distribución se encuentran también ligadas, aunque aquí se permite un cierto margen entre las dos fases (el producto, aunque perecedero, es parcialmente almacenable en el horno hasta el momento del reparto). Debido a este carácter flexible y junto al tamaño reducido de los pedidos, es admisible atender varias peticiones en un mismo viaje. Las restricciones que se tienen en cuenta para ello son las relativas a los plazos de entrega.

Analizando las características de los procesos de fabricación y distribución de hormigón y de comida a domicilio, las diferencias básicas son las siguientes:

- Los sistemas de reparto de hormigón trabajan bajo demanda determinista, debido fundamentalmente a que la recepción y descarga del hormigón en la obra forman parte de una planificación de la actividad en la obra realizada de antemano por el cliente. Los sistemas de comida rápida trabajan, en cambio, con demanda dinámica.

- Las características físicas del hormigón obligan a establecer una planificación en la que cada pedido debe ser distribuido inmediatamente a su proceso de producción. En el reparto de comida rápida es posible establecer un margen temporal pequeño entre las fases de producción y distribución.

- Los diferentes tipos de hormigón y el soporte único existente por vehículo, obligan a que la distribución de los pedidos se realice de forma unitaria. Sin embargo, el reparto de comida rápida permite la carga de varios pedidos por vehículo lo que posibilita servir varias entregas en un mismo viaje. Ello le confiere al proceso de distribución una complejidad superior, encaminada al diseño de rutas para vehículos con restricciones de tiempo.

2.2. DESCRIPCIÓN DEL PROBLEMA

En esta tesis se presenta el problema asociado a la planificación conjunta de la fabricación y distribución de pedidos (PDP), en sistemas en los que el suministro del producto está acoplado con el sistema productivo debido a la ausencia de inventarios. El problema surge en productos que no pueden ser fabricados con mucha antelación al momento exacto de ser servidos, debido a su carácter no almacenable. La fabricación de estos productos suele ser por ello un proceso bastante básico donde, en ocasiones incluso, el vehículo encargado de servir el pedido forma parte del proceso productivo.

Atendiendo a la logística reflejada en la distribución del hormigón, en este trabajo se presenta el estudio del problema sobre diferentes escenarios. En cada escenario se presenta un enfoque diferente del problema.

A continuación se detallan las características comunes en todos los escenarios del problema:

- **Demanda**

  Partimos de una situación totalmente determinista, con conocimiento pleno de los pedidos solicitados en un horizonte temporal determinado, el cual se asume como un día.

  Además de los tiempos de producción y distribución, los datos característicos de cada pedido son un valor económico que se obtiene por servir el pedido y una fecha de entrega, ya sea fija o variable.

- **Fabricación de pedidos**

La fabricación de un pedido se limita a un único proceso que se realiza sin interrupciones (*no preemption*). El tiempo de producción de cada pedido es proporcional al tamaño del mismo.

La planta de fabricación mide su capacidad de producción en número de pedidos que pueden ser atendidos en un mismo instante de tiempo. Cada pedido utiliza una unidad de capacidad durante el tiempo de fabricación, por lo que, suponiendo C unidades de capacidad en la planta, no pueden fabricarse más de C pedidos al mismo tiempo. Es decir, la fabricación de un pedido se limita a un proceso continuo que consume una unidad de capacidad y que tarda un tiempo proporcional al tamaño del pedido. Esta va a ser una de las limitaciones que impide que se puedan servir todos los pedidos.

En el proceso de fabricación no se tienen en cuenta necesidades de materiales.

- **Distribución de pedidos**

Para la distribución se cuenta con una flota de vehículos de idénticas características.

Cuando se prepara un pedido en una planta, éste es inmediatamente distribuido(no espera) al lugar donde se localiza el cliente que cursó la petición. Una vez que se produce la recepción y descarga del pedido en dicha localización, el vehículo regresa a planta y queda disponible para otro envío. Por tanto, cada pedido es servido por un solo vehículo de forma independiente. Ello implica un análisis diferente del problema respecto a una planificación en la que pudieran servirse varios pedidos en un solo viaje, aspecto que conduciría a la fase de distribución al clásico problema de diseño de rutas para vehículos (VRP).

Al asumir en el problema que el envío de un pedido es realizado por un solo vehículo, la ruta que podría realizar el vehículo, tanto la ida como la vuelta, es conocida de antemano gracias al conocimiento que se tiene de la localización del pedido. Por ello, los datos en el problema que describen la fase de distribución de un pedido son:

- El tiempo de envío hasta la localización del pedido
- El tiempo de descarga del pedido
- El tiempo de vuelta del vehículo a la planta

A lo largo de todo el documento se suponen tiempos discretos, de forma que cada unidad representa un periodo de tiempo estimado en función de la actividad concreta del sistema.

Además de considerar vehículos idénticos, se supone que los pedidos poseen un tamaño inferior a la capacidad de los vehículos.

Inicialmente, todos los vehículos se encuentran en la planta.

La figura 2.2 representa gráficamente la actividad asociada al envío de pedidos a clientes. Se ha escenificado el problema con una planta de producción.

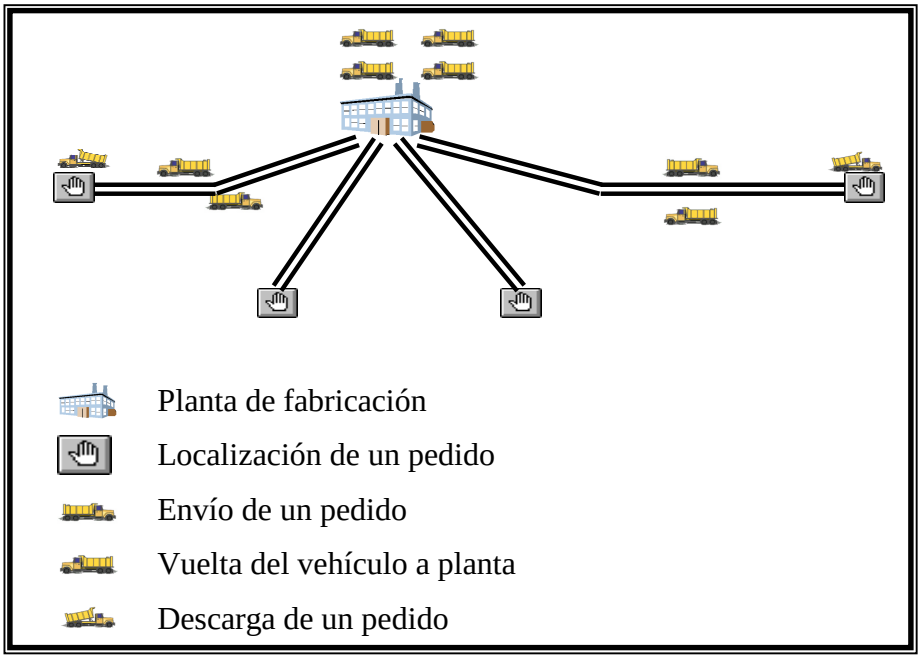


Figura 2.2: Gráfico del funcionamiento del sistema de distribución

2.3. PARÁMETROS DE ANÁLISIS Y ESCENARIOS DE ESTUDIO

El problema PDP está sujeto a diversos enfoques que dependen del entorno en el que se plantee el problema. Es posible plantear situaciones en la que las fechas de entrega impuestas por el cliente deban cumplirse con exactitud, por otras en las que se permita un cierto margen temporal para la entrega. También es posible estudiar situaciones en las que se dispone de varios centros de producción de pedidos, situados a lo largo del área geográfica de influencia, o fijar el problema con una única planta. Además de éstas, también podemos considerar diferentes enfoques en el propio proceso productivo, como que el vehículo que distribuye el pedido participe en la fabricación del mismo. Por todo ello, a continuación se detallan en primer lugar los parámetros de análisis en el problema, y en segundo lugar se

definen, en función de los valores asignados a esos parámetros, los escenarios de estudio.

2.3.1. Parámetros de análisis

A. Carácter de los Plazos de Entrega

Generalmente, en los problemas asociados al reparto de mercancías se distinguen tres casos para la gestión de las entregas:

1. El pedido puede únicamente ser entregado en el instante de entrega solicitado por el cliente.

2. El pedido puede ser entregado en cualquier instante, penalizándose las entregas respecto a un instante de entrega ideal solicitado por el cliente.

3. El pedido posee un intervalo o ventana temporal dentro del cual se considera admisible la entrega del pedido.

En nuestro problema consideraremos los casos 1 y 3, considerando, para el caso 3, un instante de entrega ideal dentro de cada ventana temporal. El caso 2 quedaría de esta forma como un caso particular del caso 3, en el que las ventanas son lo suficientemente grandes como para permitir la entrega de todos los pedidos.

B. Número de Plantas de Producción

Un factor muy importante en el estudio del problema es el número de centros para la producción de los pedidos. Distinguiremos el caso en el que la producción se encuentra totalmente centralizada en un único punto, del caso en el que se poseen varios centros de producción desde donde fabricar y enviar los pedidos.

Inicialmente, el problema se analizará a fondo teniendo en cuenta un solo centro de producción. Muchos de los resultados obtenidos de este análisis nos servirán para presentar y resolver, en la última parte del documento, el problema con varios centros de producción.

C. Participación del vehículo en el proceso productivo

Una consideración a tener en cuenta en alguno de los escenarios del problema será la participación del vehículo en el proceso productivo. En ocasiones, como en la ya comentada distribución del hormigón, el vehículo puede formar parte del proceso de fabricación del pedido, como soporte del proceso. Este caso se produciría en productos con una fabricación muy simplificada, caracterizada únicamente por un proceso como puede ser la mezcla de determinados compuestos, el cual podría realizarse directamente sobre el depósito de carga del vehículo.

Por otro lado se encuentra el caso más común en los sistemas de producción y distribución, en el que el vehículo no forma parte del proceso de fabricación, sino que sólo es necesario cuando el producto ya ha sido elaborado. Es obvio que en este caso también existe un tiempo de carga del pedido en el vehículo, que se incluye dentro del tiempo de envío.

La participación del vehículo en la fase de producción no influye en el planteamiento y resolución del problema. Por ello, este factor podría haber sido considerado en todos los escenarios, aunque en la mayoría de ellos se considera que el vehículo no participa.

D. Número de vehículos disponibles

El número de vehículos disponibles en el problema constituye, junto a la capacidad de producción, la limitación que impide la realización de todos los pedidos. En el análisis del problema plantearemos un caso en el que se relaja esta restricción, asumiendo un número ilimitado de vehículos. Esta característica simplifica en parte la resolución del problema, aunque por otro lado lleva a plantear, en un caso particular del problema, un objetivo encaminado a la minimización del número de vehículos necesarios para atender un número dado de pedidos obtenido previamente.

La siguiente tabla resume los parámetros de análisis en el problema:

Parámetro	Valores
Plazo de entrega	Instantes fijos Ventanas temporales
Número de plantas	1 planta *m* plantas
Vehículo en proceso productivo	Si No
Número de vehículos disponibles	Número fijo de vehículos Número ilimitado de vehículos

Tabla 2.1: Parámetros de análisis

2.3.2. Escenarios de estudio

Según los parámetros descritos anteriormente, el problema PDP se estudia en la primera parte del trabajo sobre dos escenarios que presuponen un único centro de producción y que varían en el tipo de plazo de entrega. A su vez, el primero plantea dos nuevos escenarios que difieren en el número de vehículos disponibles. A partir del análisis realizado sobre estos escenarios, se presentan dos nuevos escenarios del

problema con varios centros de producción de pedidos. Estos dos últimos escenarios se distinguen también en el plazo de entrega de los pedidos. A continuación se presentan las características propias de cada escenario:

- **Escenario I**

  Se estudia el problema PDP con las siguientes características:

  - Una planta de producción
  - Instantes fijos de entrega

  En el estudio de este escenario se realiza una segunda clasificación del problema atendiendo al número de vehículos disponibles. También se propone en uno de ellos la participación del vehículo en el proceso de producción.

  - **Escenario IA:**

    - Número fijo de vehículos
    - El vehículo no participa en el proceso productivo

  - **Escenario IB:**

    - Número ilimitado de vehículos
    - Participación del vehículo en el proceso productivo

- **Escenario II**

  - Una planta de producción
  - Ventanas temporales de entrega
  - El vehículo no participa en el proceso productivo
  - Número fijo de vehículos disponibles

- **Escenario III**

  - Varias plantas de producción
  - Instantes fijos de entrega
  - Número fijo de vehículos disponibles
  - El vehículo no participa en el proceso productivo

Cuando se disponen de varias plantas de producción, un pedido puede ser producido desde cualquiera o desde un subconjunto predeterminado de ellas. Una vez que un vehículo sirve un pedido, puede retornar a cualquiera de las plantas existentes.

- **Escenario IV**

  - Varias plantas de producción
  - Ventanas temporales de entrega
  - Número fijo de vehículos disponibles
  - El vehículo no participa en el proceso productivo

La siguiente tabla resume los diferentes escenarios de estudio del problema:

Escenarios		Plazo de Entrega	Vehículo en proceso productivo	Plantas de producción	Nº vehículos disponibles
I	**A**	Fijo	No	1	Fijo
	B	Fijo	Sí	1	Ilimitado
II		Variable	No	1	Fijo
III		Fijo	No	Varias	Fijo
IV		Variable	No	Varias	Fijo

Tabla 2.2: Escenarios de estudio del problema PDP

2.4. OBJETIVOS DEL PROBLEMA

Tal como hemos definido el problema PDP con un centro de producción, una representación gráfica del problema podría ser la siguiente:

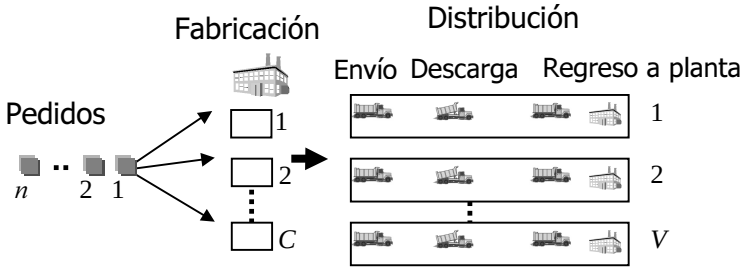


Figura 2.3: Representación esquemática del problema PDP

Partiendo de la hipótesis establecida para el problema respecto a la carga unitaria de pedidos en vehículos, podemos considerar la distribución como un proceso unitario que implicaría las tres fases de la actividad del vehículo (envío, descarga y regreso a planta). Por ello, la actividad asociada a un pedido podemos verla como un doble procesamiento secuenciado: La fase de producción y la fase de distribución

(Figura 2.2). Para la fase de producción se disponen de C unidades de producción y para la fase de distribución de V vehículos. Ello significa que los recursos para atender la actividad de los pedidos son limitados. Si a esto unimos la necesidad de respetar las fechas de entrega impuestas por el cliente, en la mayoría de los casos ocurrirá que no sea admisible atender todos los pedidos, y por tanto, el problema atenderá a seleccionar un conjunto de pedidos según un criterio determinado.

El criterio que va a establecerse en todos los escenarios del problema corresponderá con maximizar el beneficio de la selección. Para el caso de una sola planta, el beneficio depende únicamente del valor de los pedidos seleccionados, puesto que los costes de transporte son constantes en cada pedido y pueden ser incluidos dentro del propio valor del mismo. Sin embargo, en el caso de varios centros de producción, el cálculo de beneficios tiene en cuenta tanto el valor de los pedidos seleccionados como los costes de distribución originados, puesto que un pedido puede ser servido desde diferentes plantas.

Un objetivo implícito siempre en el cálculo de soluciones es el uso de un número mínimo de vehículos para atender los pedidos seleccionados. Sin embargo, este objetivo adquiere relevancia y complejidad en el supuesto de considerar un mismo valor para todos los pedidos. Es decir, maximizar el beneficio coincidiría con maximizar el número de pedidos atendidos. Con este supuesto aumenta la posibilidad de soluciones óptimas alternativas, lo que da pie a buscar aquélla que utilice el menor número de vehículos. Este caso particular se presenta en el escenario IB del problema, en el que se admite que los vehículos disponibles son ilimitados, es decir, suficientes siempre para atender los pedidos solicitados.

Por otro lado, cuando se consideran ventanas temporales el beneficio del pedido se ve modificado dependiendo del instante en el que se produce la entrega. Con ello se quiere reflejar la importancia de la calidad del servicio. La calidad del servicio en la entrega de un pedido se mide respecto a la desviación sobre el instante de entrega ideal solicitado por el cliente. Para reflejar esta característica en la función objetivo del problema, el valor del pedido sufrirá una disminución lineal progresiva conforme el instante de entrega del pedido se aleje del instante solicitado por el cliente.

2.5. NOTACIÓN BÁSICA

A continuación se describe la notación básica utilizada a lo largo del documento para todos los elementos del problema. Para una mayor claridad, se excluye la notación asociada a los escenarios con múltiples plantas de producción, dejando ésta para los capítulos correspondientes a esos escenarios.

Denotemos por $P = \{1...i...n\}$ el conjunto de n pedidos solicitados. Los datos básicos de cada pedido i son los siguientes:

- e_i : instante de entrega solicitado.

- $[A_i, B_i]$: Ventana temporal en la que se considera admisible la entrega del pedido; $e_i \in [A_i, B_i]$ (Plazo de entrega temporal).

- w_i : valor o beneficio obtenido con la entrega del pedido.

La notación de tiempos es la siguiente:

- tp_i : tiempo de fabricación del pedido

- ti_i : tiempo de envío del pedido.

- tr_i : tiempo de regreso del vehículo a la planta.

- tu_i : tiempo de recepción o descarga del pedido.

Visto de una forma gráfica:

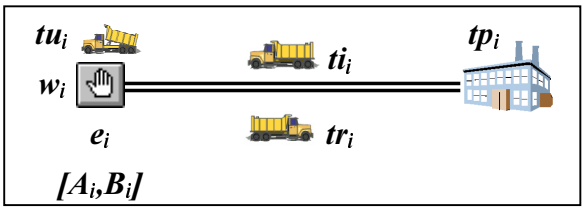


Figura 2.4: Datos del problema PDP con un centro de producción

El tiempo total de actividad del vehículo, denominado tiempo de distribución, se obtiene como la suma del tiempo de envío, descarga y vuelta del vehículo a la planta: $td_i = ti_i + tu_i + tr_i$

A partir de los datos de partida del pedido (tiempos de proceso e instante de entrega e_i), es posible el cálculo de los siguientes:

s_i : Instante de inicio de la fase de producción del pedido; $s_i = e_i - ti_i - tp_i$

l_i: Instante de inicio de la fase de fabricación del pedido; $l_i = e_i - ti_i$

f_i: Instante de finalización de la actividad asociada al pedido; $f_i = e_i + tu_i - tr_i$

En el caso de ventanas temporales de entrega $[A_i,B_i]$, es posible calcular como medida análoga, una ventana temporal de comienzo de la actividad del pedido $[a_i,b_i]$:

$a_i = A_i - ti_i - tp_i$

$b_i = B_i - ti_i - tp_i$

De forma general, la actividad asociada a la fabricación y distribución de un pedido correspondería con la siguiente figura:

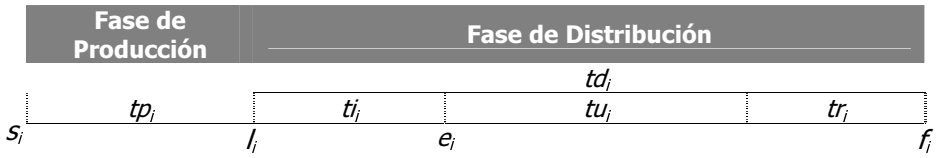


Figura 2.5: Datos de la actividad de un pedido

Según la figura 2.5 y como ya se ha comentado, la realización de un pedido, entendiendo realización como la actividad completa del mismo, está dividida en dos fases diferenciadas, fabricación y distribución. Estas fases deben realizarse de forma ininterrumpida, es decir, sin esperas. Dentro de cada una de las fases tampoco se permiten interrupciones.

Además de los datos señalados para los pedidos, se definen:

- C : Capacidad de producción de la planta
- V : Número total de vehículos disponibles

2.6. ILUSTRACIÓN

Para una mejor comprensión de la problemática asociada al problema PDP, se presentan algunos ejemplos de planificación en los escenarios I y II del problema.

- **ESCENARIO I:** Producción y distribución conjunta de pedidos con instantes fijos de entrega y una planta de producción

La tabla 2.3 muestra los datos correspondientes a un conjunto de pedidos solicitados.

Pedidos	A	B	C	D	E	F	G	H	I
e_i	5	7	6	11	15	4	19	21	21
w_i	10	15	10	5	10	9	6	11	6
tp_i	2	2	2	1	2	1	2	2	1
ti_i	2	3	3	2	2	1	2	2	1
tu_i	1	2	2	1	1	1	1	1	1
tr_i	2	3	4	2	2	2	1	2	1

Tabla 2.3: Datos de ejemplo para Escenario I

La capacidad de producción en la planta es $C = 1$ y el número de vehículos es $V = 3$. Para una visión más clara de la actividad en el tiempo de los pedidos utilizamos un diagrama cronológico donde cada pedido se representa de la siguiente forma:

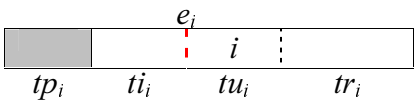


Figura 2.6: Representación de un pedido en el Escenario I

Según ello, el diagrama temporal asociado con los pedidos de la tabla 2.3 sería el siguiente:

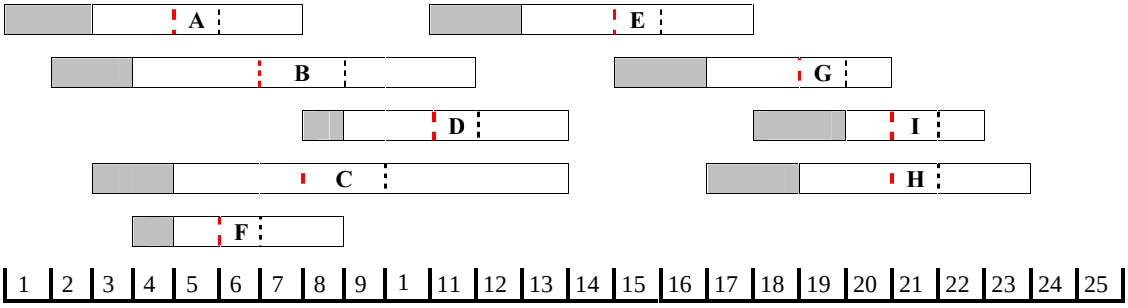


Figura 2.7: Diagrama temporal del ejemplo de tabla 2.3

Supuesta una capacidad de fabricación de C = 1, no podrían fabricarse más de 1 pedido al mismo tiempo en la planta, por lo que los pares de pedidos A y B así como el par B y C se verían afectados por esta limitación (figura 2.8). Debido a que no se permiten interrupciones en los procesos y además el plazo de entrega es fijo, habría que descartar la fabricación de alguno de ellos. En el ejemplo también aparecen solapados los procesos de fabricación de los pedidos C y F y los de H e I.

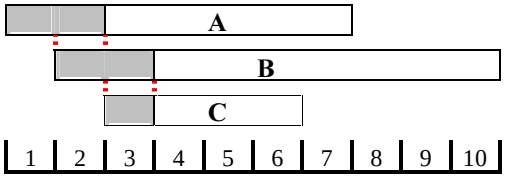


Figura 2.8: Solapamiento entre pedidos

En las figuras 2.9 y 2.10 se muestran dos planificaciones admisibles supuesto que los vehículos no formen parte del proceso de fabricación.

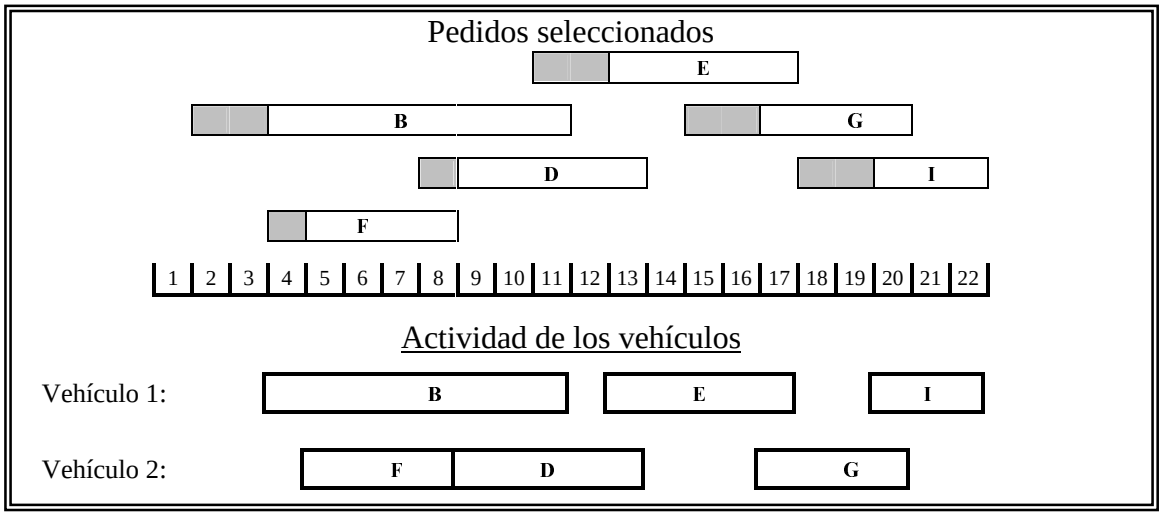


Figura 2.9: Diagramas de la planificación I1

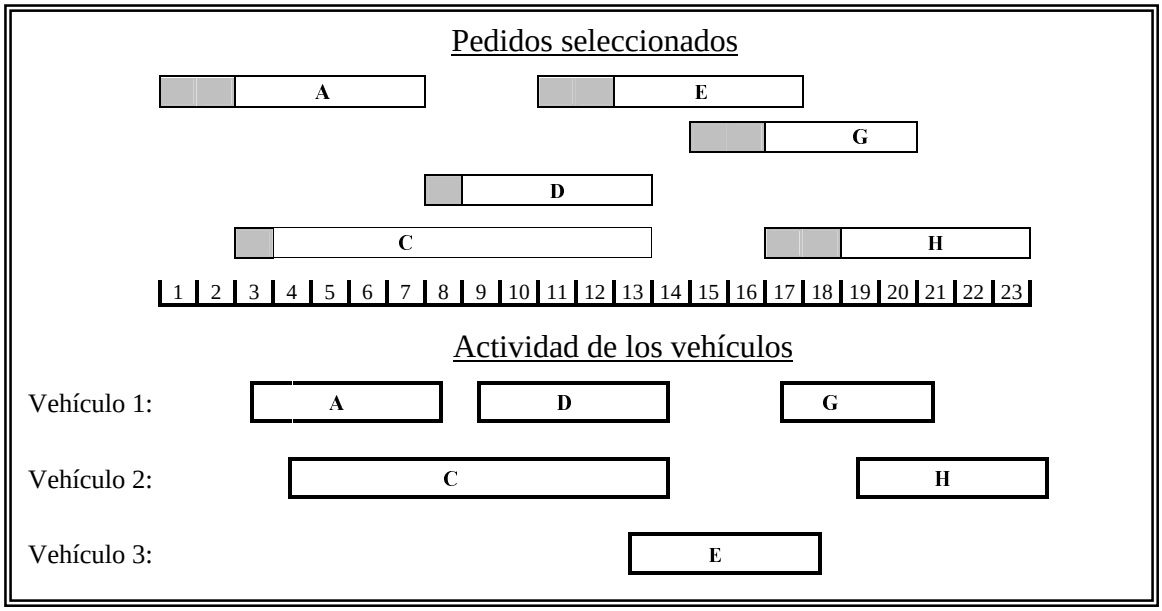


Figura 2.10: Diagramas de la planificación I2

La planificación I1 necesita únicamente dos vehículos. El tercer vehículo permanece inactivo. La planificación I2 requiere sin embargo el uso de los tres vehículos. En los dos casos, se ha hecho uso del número mínimo de vehículos necesarios para atender los pedidos seleccionados. Veamos los datos de cada solución:

Planificación	Número de pedidos	Beneficio	Vehículos utilizados
1	6	51	2
2	6	52	3

Tabla 2.4: Comparación de planificaciones

La planificación I2 presenta un beneficio mayor que la planificación I1. Sería la planificación elegida en el caso del Escenario IA del problema.

Sin embargo, si hubiéramos supuesto el mismo valor para todos los pedidos, maximizar el beneficio se convertiría en maximizar el número de pedidos. En ese caso, las dos soluciones presentan el mismo resultado (6 pedidos). Planteando un segundo objetivo que minimice el número de vehículos utilizados, la planificación I1 sería la más eficiente, pues únicamente usa dos vehículos (Escenario IB).

Como veremos en el capítulo dedicado al estudio del Escenario I, además de obtener planificaciones a partir de una capacidad de producción C y un número vehículos V, es factible también resolver cuestiones como qué capacidad de fabricación y número de vehículos harían falta para satisfacer toda la demanda.

- **ESCENARIO II:** Producción y distribución conjunta de pedidos con ventanas temporales de entrega y una planta de producción

Aunque el estudio relacionado con la minimización del número de vehículos se ha centrado exclusivamente en el Escenario I del problema, para el caso de ventanas temporales pueden plantearse las mismas cuestiones vistas para el caso de entregas fijas. Veamos un ejemplo: Los datos de cada pedido son los siguientes:

Pedidos	A_i	e_i	B_i	tp_i	ti_i	tu_i	tr_i	w_i
A	11	12	13	2	7	2	7	5
B	7	8	8	2	2	1	1	4
C	10	10	10	1	5	3	5	8
D	19	19	20	1	2	3	2	5

Tabla 2.5: Datos de ejemplo para Escenario II

Cada pedido se representa de la siguiente forma:

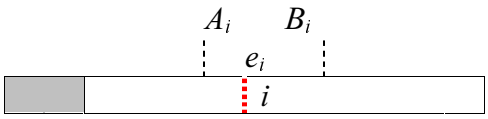


Figura 2.11: Representación de un pedido en el Escenario II

A partir de la representación de los pedidos de la figura 2.11, el diagrama temporal para el ejemplo sería el siguiente:

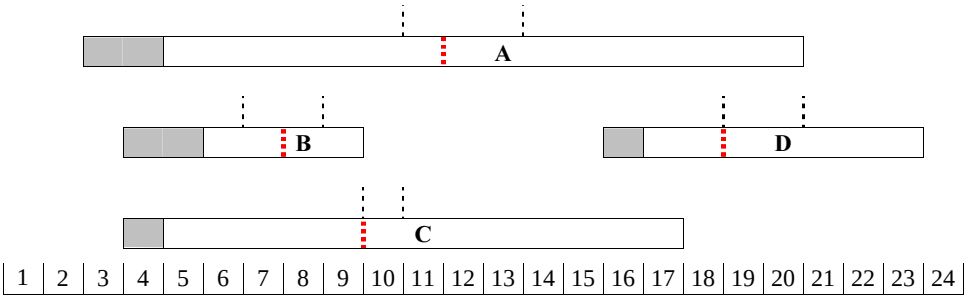


Figura 2.12: Diagrama temporal del ejemplo de tabla 2.5

Supuesta una capacidad de fabricación $C = 1$ y un número de vehículos $V = 3$, veamos 3 planificaciones (II1, II2 y II3) admisibles para los pedidos:

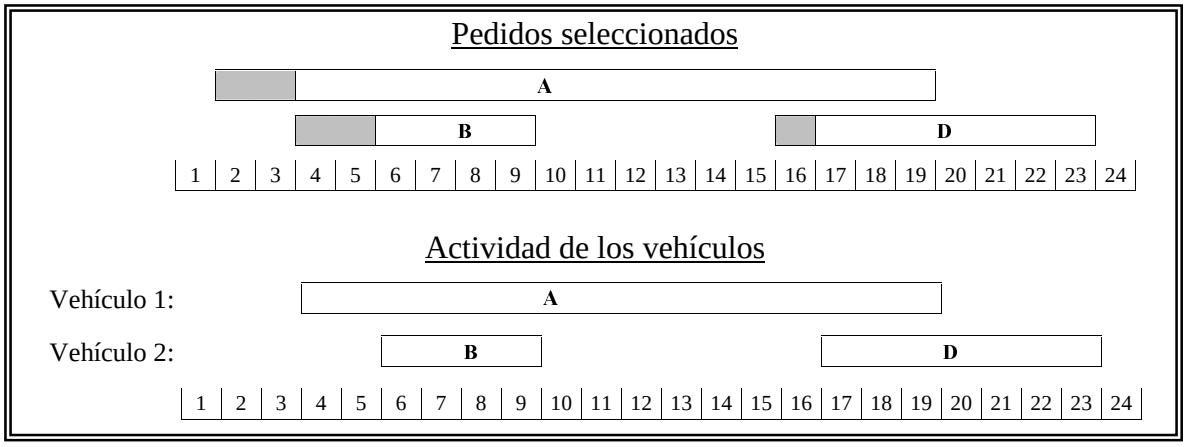


Figura 2.13: Diagramas de la planificación II1

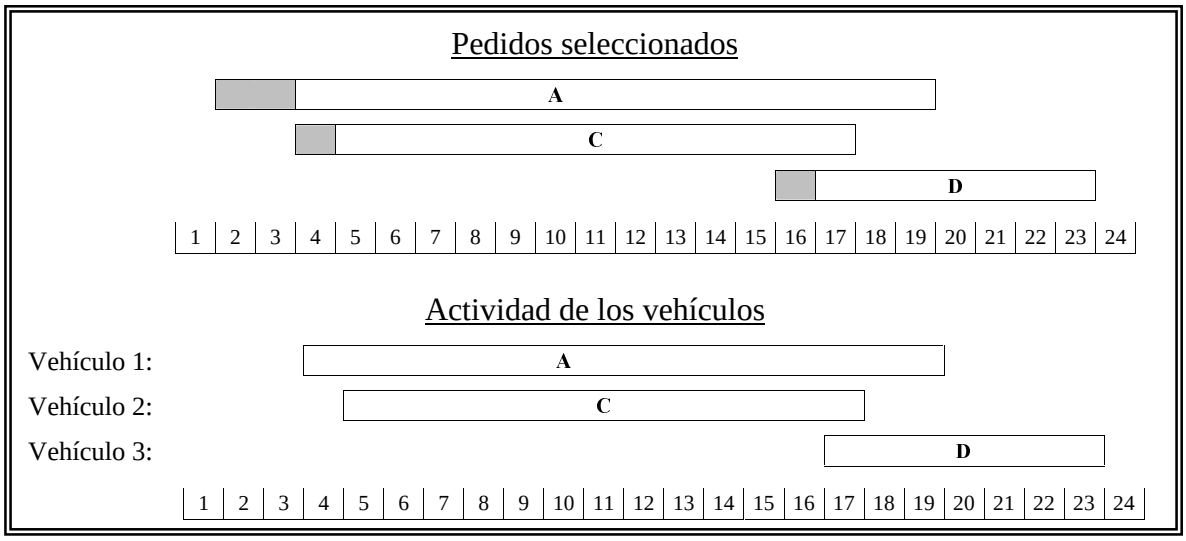


Figura 2.14: Diagramas de planificación II2

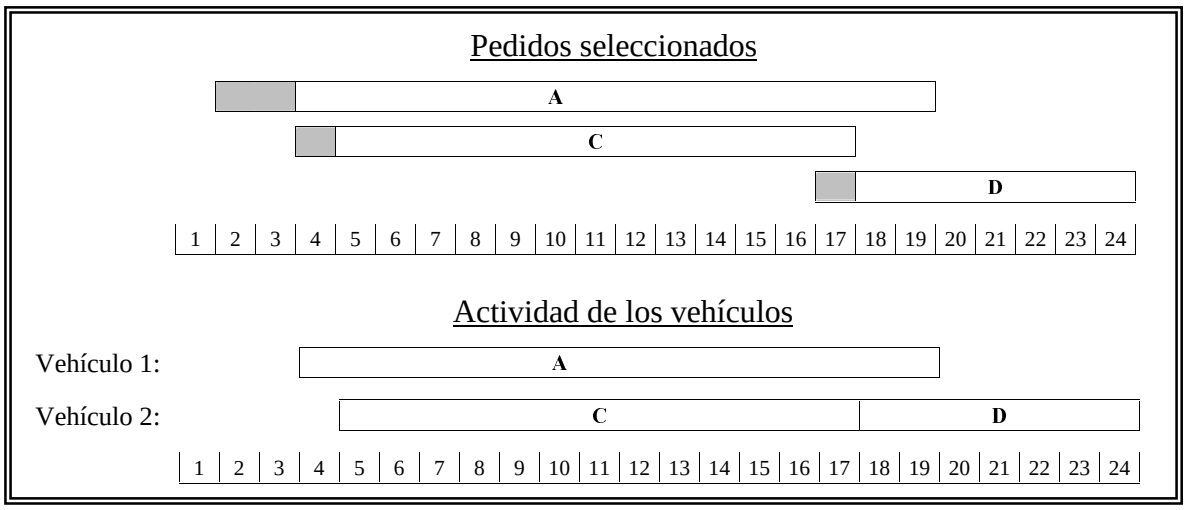


Figura 2.15: Diagramas de la planificación II3

Atendiendo a los datos resumen de las tres planificaciones (tabla 2.6), las planificaciones II2 y II3 mejoran a la planificación II1 si tenemos en cuenta el valor

de los pedidos servidos. Sin embargo, si hubiéramos tenido en cuenta además las desviaciones respecto a la fecha de entrega, la planificación II2 resulta mejor que la II3. Por otro lado, la planificación II2 hace uso de un vehículo más que la planificación II3.

Planificación	II1	II2	II3
Número de pedidos servidos	3	3	3
Valor total de los pedidos	14	18	18
Número de pedidos entregados puntualmente	2	2	1
Número de pedidos adelantados	1	1	1
Número de pedidos retrasados	0	0	1
Instantes adelantados	0	0	1
Instantes retrasados	1	1	1
Vehículos utilizados	2	3	2

Tabla 2.6: Comparación de planificaciones

La optimización del problema PDP en el Escenario II atenderá al valor de los pedidos teniendo en cuenta la aplicación de penalizaciones en el caso de retraso o adelanto de los pedidos sobres su fecha ideal de entrega.

Capítulo 3

Estado actual del problema

Índice

3.1. INTRODUCCIÓN

En este capítulo se analizan las características del procesamiento al que se ven sometidos los pedidos en el sistema, con objeto de encuadrar el problema en el campo de la secuenciación de trabajos en máquinas.

Para ello, en primer lugar se introducen los conceptos generales asociados a la secuenciación de trabajos, para posteriormente determinar las analogías con diversos entornos de procesamiento de trabajos.

En el último apartado del capítulo se describen las técnicas de resolución empleadas para el problema PDP.

3.2. SECUENCIACIÓN DE TRABAJOS EN MÁQUINAS

La secuenciación de trabajos en máquinas, universalmente conocido como *scheduling*, se puede definir como la asignación en el tiempo de los recursos disponibles con objeto de optimizar una determinada medida de comportamiento. Más concretamente, y a partir de un determinado criterio, se trata de establecer la secuencia para el procesamiento de una serie de trabajos sobre un conjunto de máquinas [Bertolissi, 2000].

El gran número de publicaciones existentes relativas a problemas de secuenciación se debe al amplio espectro de características que pueden asociarse a los trabajos y al modo de procesamiento en el sistema. Básicamente, un problema viene determinado por el entorno asociado a las máquinas, las características de los trabajos y el criterio de optimización. Los problemas de secuenciación poseen una aplicación inmediata en la mayoría de los sistemas de producción, en algunos sistemas de transporte y distribución y en determinadas actividades del sector servicio.

3.2.1. Tipos de entorno

Respecto a la arquitectura del sistema en relación con la disposición de las máquinas, se distinguen en una primera clasificación, los sistemas con máquinas en serie y máquinas en paralelo. En los entornos con máquinas en serie, cada máquina realiza una operación diferente, del conjunto de operaciones a las que se someten los trabajos. Cuando hablamos de máquinas en paralelo, se entiende que una determinada operación sobre un trabajo puede ser procesada sobre una cualquiera de entre un conjunto de máquinas. La figura 3.1 ilustra las dos arquitecturas:

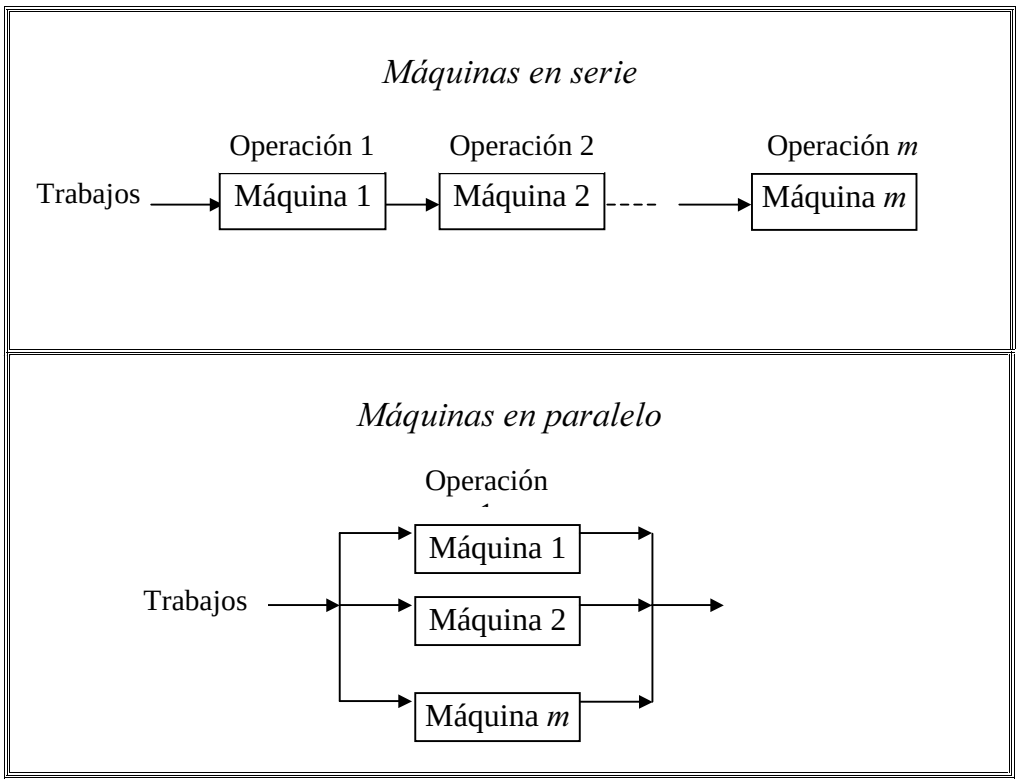


Figura 3.1: Arquitectura de máquinas en serie y máquinas en paralelo

Los entornos de máquinas en serie se clasifican en función del modelo o esquema de paso de los trabajos por las diferentes máquinas [Eguia, 1996]:

- Sistemas de flujo uniforme (*Flow shop*): El modelo de paso es el mismo para todos los trabajos. Todos los trabajos pasan por cada una de las máquinas del sistema usando el mismo orden de paso por las mismas.

- Sistemas de tipo taller (*Job shop*): Cada trabajo tiene su propio esquema de paso por las máquinas.

- Sistemas de taller abierto (*Open shop*): El modelo de paso de cada trabajo por las máquinas es libre.

Respecto a los sistemas de máquinas en paralelo se distinguen tres tipos:

- Máquinas idénticas: El tiempo de proceso de una operación es idéntico en cada máquina.

- Máquinas uniformes: Cada máquina posee una velocidad de proceso diferente, independiente de los trabajos.

- Máquinas no relacionadas: Cada máquina posee una velocidad de proceso diferente sobre cada trabajo.

Además de los sistemas con máquinas en serie y máquinas en paralelo, también se distingue un sistema híbrido denominado sistema de flujo uniforme con máquinas en paralelo (*Flexible Flow Shop*).

- **Sistemas de flujo uniforme con máquinas en paralelo**

El término taller de flujo uniforme con máquinas en paralelo (*flow shop with parallel machines*, FSPM), que también puede aparecer como taller de flujo uniforme con múltiples procesadores (*flow shop with multiple processors*), taller híbrido de flujo (*hybrid flow shop*), taller uniforme de permutación (*permutation flow shop*) o, simplemente, taller flexible de flujo uniforme (*flexible flow shop*), es una generalización de los entornos *flow shop* y máquinas en paralelo. En vez de *m* máquinas en serie, existen *m* centros o estaciones de máquinas en serie, cada uno de ellas formado por un conjunto de máquinas en paralelo (figura 3.2). Cada trabajo tiene que ser procesado primero en la estación 1, después en la estación 2, y así sucesivamente hasta la última estación. En cada estación el trabajo requiere sólo una máquina y cualquier máquina puede procesar cualquier trabajo.

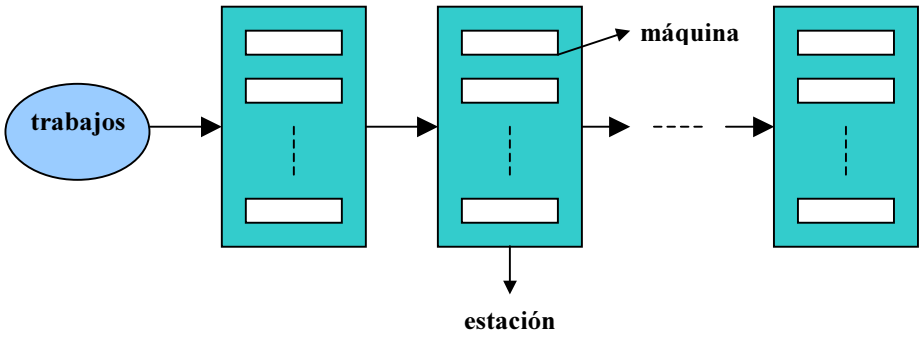


Figura 3.2: Arquitectura de un sistema de flujo uniforme con máquinas en paralelo

Posteriormente se alude a los sistemas FSPM con ciertas características específicas de procesamiento, ya que dichos sistemas representan la arquitectura básica sobre la que se presenta el problema PDP.

3.2.2. Notación y representación

Existen varias formas de representar un problema de secuenciación. Una de las más comúnmente usadas se describe por una tripleta $\alpha/\beta/\gamma$[Graham et al, 1979]. El campo α indica el entorno de las máquinas (número de máquinas y tipo de sistema). El campo β describe las características y restricciones de procesamiento de los trabajos. El campo γespecifica el objetivo del problema.

Tanto las características de los trabajos como las restricciones de proceso y los tipos de sistemas, constituyen una amplísima variedad de datos, que no es fácil

generalizar, sobre todo por la multitud de trabajos que existen en este campo. Por ello, a continuación se exponen las características más relevantes y que posteriormente ayudarán a la definición de algunos escenarios para el problema PDP. La terminología utilizada hace referencia a la notación descrita en [Cheng, 1990], [Hall y Sriskandarajah, 1996] y [Pinedo, 1995].

Terminología para los entornos de máquinas (α)

F_m: Sistema de flujo uniforme con m máquinas en serie

J_m: Sistema de taller con m máquinas en serie

O_m: Sistema de taller abierto con m máquinas en serie

P_m: m máquinas idénticas en paralelo

Q_m: m máquinas uniformes en paralelo

R_m: m máquinas no relacionadas en paralelo

S_m: Sistema de flujo uniforme con m centros de máquinas en paralelo

Terminología asociada a los trabajos (β)

- DATOS

  $J = \{J_1, J_2,.., J_i, .., J_n\}$: Conjunto de n trabajos;

  $M = \{M_1, M_2,.., M_j, ..., M_m\}$: Conjunto de m máquinas;

  t_{ij} : Tiempo de proceso del trabajo J_i en la máquina M_j.

  r_i : Instante de llegada de J_i al sistema. Cuando todos los trabajos están disponibles al comienzo del procesamiento, $r_i = 0$.

  d_i : Fecha de entrega del trabajo J_i.

  w_i : Peso del trabajo J_i.

- VARIABLES

  C_i : Tiempo de finalización de J_i.

  $F_i = C_i - r_i$: Tiempo de permanencia del trabajo en el sistema.

  $L_i = C_i - d_i$: Mide la desviación respecto a la fecha de entrega. Si $L_i<0$ (retraso negativo), $|L_i|$ representa las unidades de adelanto.

T_i = máximo {0, L_i}= máximo {0, $C_i - d_i$}: Tardanza de J_i o número de instantes de retraso de J_i;

E_i = máximo {0, $d_i - C_i$}: Número de instantes de adelanto de J_i.

Variables booleanas para el control de los trabajos que se retrasan y adelantan:

$$X_i^T = \begin{cases} 1 \text{ si } C_i > d_i \\ 0 \text{ en otro caso} \end{cases}$$

$$X_i^E = \begin{cases} 1 \text{ si } C_i < d_i \\ 0 \text{ en otro caso} \end{cases}$$

De manera análoga, se pueden definir variables booleanas para el control de los trabajos finalizados sin retraso ni adelanto:

$$X_i = \begin{cases} 1 \text{ si } C_i = d_i \\ 0 \text{ en otro caso} \end{cases}$$

3.2.3. Restricciones de procesamiento de los trabajos

En este apartado destacamos determinadas características que pueden presentarse en el procesamiento de los trabajos:

- *Preemption*: Rotura de trabajos. Esta característica hace referencia a la posibilidad de abandonar el procesamiento de un trabajo en una máquina sin haber concluido la operación, regresando más tarde para finalizarla.

- Restricciones de precedencia: En algunos entornos aparecen relaciones de precedencia obligada entre pares de trabajos.

- *No-wait* (No espera): Aparece en entornos de máquinas en serie en los que los trabajos deben ser procesados desde su inicio en la primera máquina, hasta su finalización en la última máquina, sin ninguna interrupción entre máquinas.

- *Blocking* (Bloqueo): Esta característica aparece en sistemas de fabricación en serie en los que no están permitidos los *buffers* intermedios de un cierto tamaño entre máquinas, puesto que bloquean el funcionamiento de las mismas.

3.2.4. Objetivos

Los objetivos o criterios para la búsqueda de soluciones se pueden agrupar en tres grandes grupos [Rinnoy Kan, 1976]:

1. Criterios basados en tiempos de finalización de los trabajos:

ΣC_i : Minimizar la suma de los tiempos de finalización de los trabajos.

$\Sigma w_i C_i$: Minimizar el coste total asociado a la finalización de los trabajos. El peso w_i se entiende como un coste de espera o un valor añadido al trabajo J_i.

$C_{max} = \text{Max}\{C_1,\ldots,C_n\}$: Minimizar el tiempo de finalización de todos los trabajos, también llamado longitud de la programación (*makespan*).

2. Criterios basados en fechas de entregas:

ΣL_i : Minimizar la suma de retrasos o retraso total. Equivalente a minimizar el retraso medio. De forma análoga al grupo anterior se estudian $\Sigma w_i L_i$ *y* L_{max}.

ΣT_i : Minimizar la tardanza total. De forma análoga se definen $\Sigma w_i T_i$ *y* T_{max}.

$\Sigma(E_i + T_i)$: Minimizar la suma de las desviaciones de los instantes de finalización de los trabajos respecto a sus fechas de entrega d_i. Cuando se establecen penaltis u_i para los adelantos y v_i para los retrasos de cada trabajo, el objetivo viene a ser $\Sigma(u_i E_i + v_i T_i)$.

ΣX^T_i : Minimizar el número de trabajos retrasados. También se estudia la minimización del coste de los trabajos retrasados, representado por $\Sigma w_i X^T_i$.

$\Sigma(X^T_i + X^E_i)$: Minimizar el número de trabajos adelantados y retrasados. Igual que anteriormente, también se plantea el criterio $\Sigma w_i(X^T_i + X^E_i)$. Este objetivo es equivalente al de maximizar los trabajos terminados justo en su fecha de entrega (*on time*) (Max ΣX_i o bien Max $\Sigma w_i X_i$).

3. Criterios basados en costes de inventario y utilización de máquinas

ΣI_j : Minimizar el tiempo total en que están desocupadas las máquinas, siendo $I_j = C_{max} - \Sigma t_{ij}$, y Σt_{ij} la suma de los tiempos de procesado de todos los trabajos sobre la máquina M_j.

$\Sigma v_j I_j$: Minimizar el tiempo ponderado de desocupación de las máquinas, siendo v_j un peso por unidad de operación.

3.3. EL PROBLEMA PDP EN EL CAMPO DE LA SECUENCIACIÓN DE TRABAJOS

Tal como se ha definido el problema PDP con un centro de producción, en la fase de producción se cuentan con *C* unidades de capacidad, que equivalen a un máximo de *C* pedidos simultáneos en cualquier instante de tiempo. Para la fase de distribución se disponen de *V* unidades de distribución, es decir, un máximo de *V* vehículos distribuyendo pedidos al mismo tiempo. Por tanto, el procesamiento completo de un pedido conlleva dos fases, cada una formada por un número limitado de recursos (figura 3.3). Además, debido al carácter no almacenable del producto, estamos obligando a que el paso de una fase a otra se produzca de manera inmediata, sin tiempos de espera.

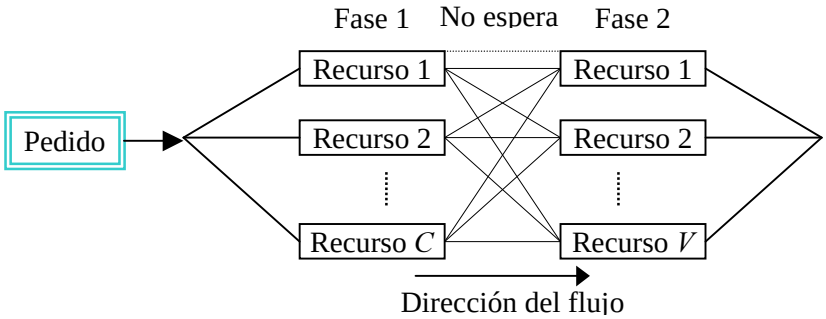


Figura 3.3: Arquitectura del problema PDP

La arquitectura representada en la figura 3.3 es equivalente a la de un sistema de flujo uniforme con máquinas en paralelo (FSPM) con las siguientes características:

- Dos centros de máquinas: La operación en el primer centro correspondería con la producción del pedido. El segundo centro sería la distribución del pedido. El centro de producción estaría formado por *C* recursos o máquinas y el centro de distribución por *V* máquinas.

- Los pedidos corresponderían con los trabajos a procesar.

- Existe restricción de *no-wait* (no espera) en el procesamiento de los trabajos por los centros de máquinas.

Con todo ello, hemos asimilado la problemática asociada al problema PDP con un entorno concreto dentro del campo de la secuenciación de trabajos. Por otro lado, si asumimos en el problema que el número de recursos en alguna de las dos fases es lo suficientemente grande para poder atender todos los pedidos, la arquitectura del problema queda simplificada a una sola fase de procesamiento. Ello sería equivalente a la arquitectura de un sistema con máquinas en paralelo (figura 3.4).

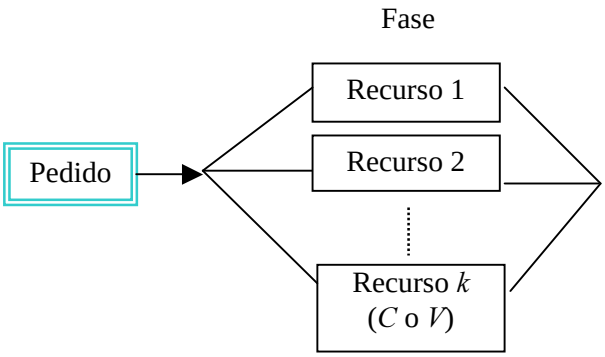


Figura 3.4: Arquitectura para un solo recurso

Otros parámetros del problema dependen de la distinción entre fechas fijas de entrega o plazo de entrega temporal. Veamos por ello las características propias del problema PDP en cada uno de los escenarios.

Características propias de cada escenario

- **Escenario IA**

  - Todos los trabajos poseen fecha de llegada al sistema $r_i = s_i$ (instante de comienzo de la fase producción).

  - La fecha de entrega del trabajo correspondería con el instante de finalización de la fase distribución, definida como f_i ($d_i = f_i$).

  - Entorno FSPM con dos centros de máquinas y *no-wait* en proceso.

  - El objetivo del problema es la maximización del beneficio de la planificación, es decir, de los pesos de los trabajos servidos. Este criterio es equivalente a considerar la maximización de los pesos de los trabajos finalizados en su fecha de entrega (Max $\Sigma w_i X_i$).

  - Atendiendo a la notación definida en el capítulo, el problema en el Escenario I, correspondería con la siguiente tripleta:

$$S_2 / r_i, \textit{no-wait} / \text{Max } \Sigma w_i X_i$$

- **Escenario IB**

  - $r_i = s_i$

- La única limitación existente en esta fase es la capacidad de producción. El problema no tiene en cuenta, por tanto, la fase de distribución del producto, lo que significa que la fecha de entrega es el fin de la fase de producción:

  $d_i = l_i - 1$ (l_i = instante de comienzo de la fase de distribución).

- Sistema de máquinas idénticas en paralelo.

- Objetivo: Max $\Sigma w_i X_i$

$$P_C / r_i / \text{Max } \Sigma w_i X_i$$

- **Escenario II**

Cuando existen ventanas temporales para la entrega del pedido existe una cierta holgura en el intervalo para el procesamiento de los trabajos. Las características para este escenario son las siguientes:

- $r_i = a_i$ (primer instante en el que podría comenzar el procesamiento del pedido).

- La fecha de entrega del trabajo correspondería con el instante de finalización de la fase distribución para la fecha ideal solicitada por el cliente ($d_i = f_i$). Los retrasos y adelantos pueden medirse sobre las desviaciones respecto a f_i de igual forma que sobre el instante ideal de entrega e_i.

- Entorno FSPM con dos centros de máquinas y *no-wait* en proceso.

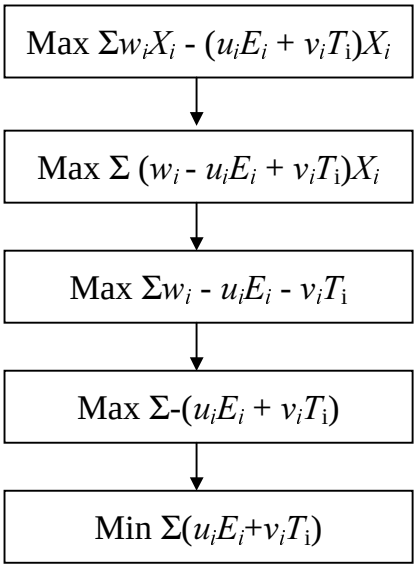
- El objetivo del problema PDP es la maximización del beneficio de la planificación teniendo en cuenta la calidad del servicio. A partir de los objetivos generales definidos en el campo de la secuenciación de trabajos en entornos *just-in-time*, el objetivo en este escenario corresponde con la siguiente expresión: [Max $\Sigma w_i X_i$ - $(u_i E_i + v_i T_i) X_i$]

El problema quedaría definido como:

$$S_2 / r_i, \textit{no-wait} / \text{Max } \Sigma w_i X_i - (u_i E_i + v_i T_i) X_i$$

En el caso de permitir ventanas temporales muy amplias, la limitación del número de recursos únicamente impondría una disminución de la calidad del servicio, pero no limitaría la selección de pedidos puesto que los pedidos podrían ser servidos sobre un amplio margen del horizonte temporal. El objetivo en ese caso se

simplificaría, pues todos los X_i tomarían valor 1, quedando como criterio la minimización de las penalizaciones por retraso y adelanto de los pedidos respecto a su fecha de entrega:



La definición del problema quedaría entonces como:

$$S_2 / r_i, \textit{no-wait} / \Sigma(u_i E_i + v_i T_i)$$

- **Escenarios III y IV**

Los escenarios III y IV definen el problema PDP bajo un entorno difícilmente adaptable a un sistema de programación de trabajos en máquinas. El problema en estos escenarios posee un entorno específico basado también en asignación de recursos en el tiempo, pero con una diferencia importante respecto al entorno FSPM de los escenarios anteriores: En el segundo centro de máquinas, es decir, en la fase de distribución, la disponibilidad de las máquinas en un instante dado, depende tanto del recurso que se haya utilizado en la fase de producción, como de la programación de trabajos realizada hasta ese instante.

La figura 3.5 representa la arquitectura de un sistema con varias plantas. La fase de producción multiplica sus recursos debido a la presencia de varias plantas. Cada planta $k$poseería un número de recursos de producción igual a su capacidad C_k.

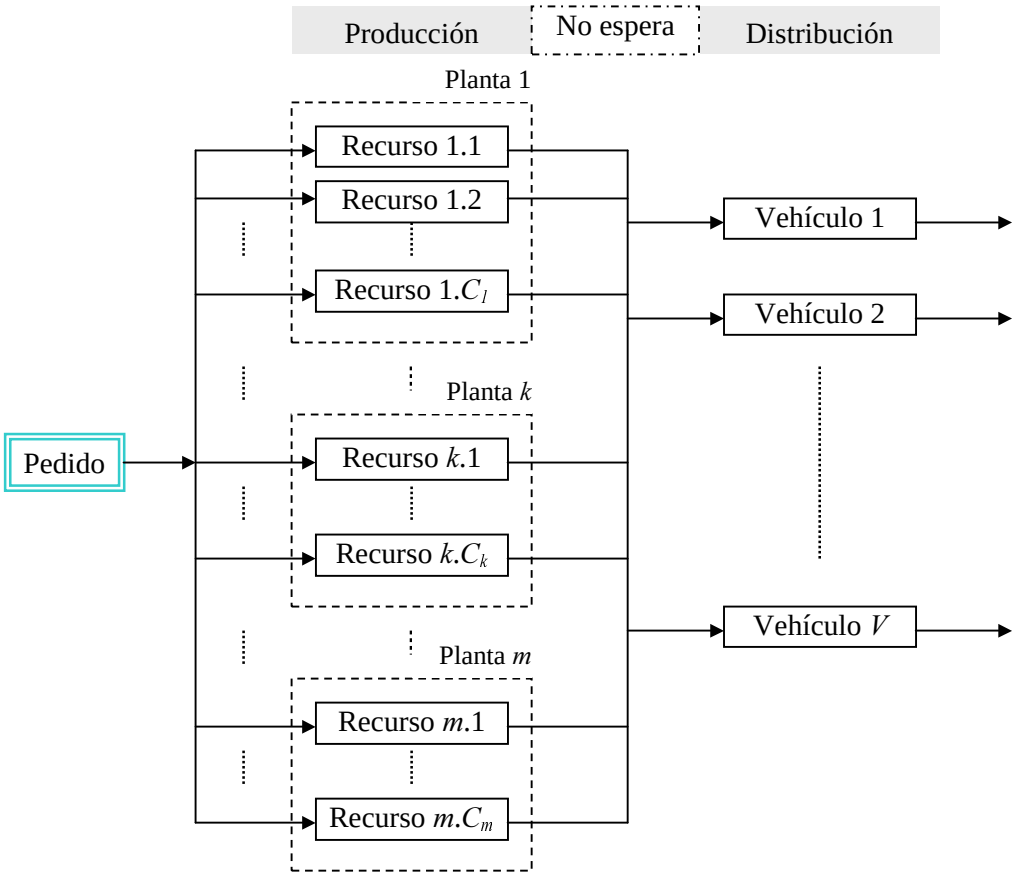


Figura 3.5: Arquitectura para el problema PDP con varias plantas

El problema aparece en la fase de distribución, donde se cuentan con V vehículos, es decir, V recursos o máquinas. Un pedido que ha hecho uso de un recurso de producción en la planta k, sólo podría utilizar un recurso de distribución que estuviera en la planta k (una vez que un vehículo sirve un pedido, puede regresar a cualquiera de las plantas). Esto significa que el número de recursos con los que puede contar un pedido en la fase de distribución se determina dinámicamente. Esto caracteriza un problema de asignación de recursos que no se acomoda a ningún entorno clásico de la secuenciación de trabajos en máquinas.

Sin embargo, partes del problema, como la determinación de admisibilidad en la asignación de trabajos a plantas, poseen una estructura semejante a la reflejada en alguno de los problemas de secuenciación que se describen en la siguiente sección.

3.4. REVISIÓN BIBLIOGRÁFICA

En este apartado se presenta una revisión, dentro del campo de la secuenciación de trabajos en máquinas, de las características comentadas para el problema PDP. Según ello, se tratarán los siguientes temas:

- Secuenciación en máquinas en paralelo de trabajos con intervalo fijo de proceso: Dentro del amplio espectro de problemas de secuenciación en máquinas en paralelo, nos centramos en aquellos en los que los trabajos poseen un intervalo [*a, b*] dentro del que deben de ser procesados.

- Secuenciación de trabajos en sistemas *just-in-time*: La secuenciación de trabajos *just-in-time* hace referencia a la importancia de la finalización de los trabajos en su fecha de entrega, asignando la misma importancia a los retrasos como a los adelantos.

- Secuenciación de trabajos en sistemas de flujo uniforme con máquinas en paralelo (FSPM) y sistemas FSPM con *no-wait* en proceso

3.4.1. Secuenciación de trabajos con intervalos fijos de proceso

Generalmente, los problemas de secuenciación de trabajos en máquinas están íntimamente ligados al problema de la programación de la producción en sistemas productivos. La secuenciación se sitúa en la última fase de este proceso y se reduce a ordenar, en un horizonte temporal determinado, las órdenes de producción calculadas de antemano según unas necesidades de producción. Por ello, resulta obvio que el conjunto de tareas a secuenciar deben ser realizadas por completo, centrándose la problemática en el establecimiento del programa de paso de los trabajos por las máquinas de forma que se optimice alguno de los criterios señalados con anterioridad. En el establecimiento de dicho programa se suele considerar admisible el procesamiento de cualquier trabajo en cualquier instante del horizonte temporal. Sin embargo, algunos investigadores han fijado su atención en problemas en los que cada uno de los trabajos posee un intervalo fijo dentro del cual debe ser procesado por completo, no permitiendo, por tanto, el procesamiento de los trabajos en cualquier instante. Asumir esta restricción puede conllevar, cuando los recursos son limitados, a que no todos los trabajos puedan ser llevados a cabo.

Una posible clasificación de los problemas con intervalo de procesamiento atendería al propio tamaño del intervalo:

a) Intervalo de procesamiento = Tiempo de procesamiento del trabajo

En este caso los trabajos deben de comenzar su procesamiento justo cuando llegan al sistema. Si llamamos $[a_i,b_i]$ al intervalo de procesamiento, entonces $a_i = r_i$ (instante de llegada del trabajo al sistema), $b_i = d_i$ (fecha de entrega) con $b_i - a_i = t_i$ (tiempo de proceso del trabajo).

b) Intervalo de procesamiento > Tiempo de procesamiento del trabajo

Los trabajos disponen de una cierta holgura dentro de su intervalo de procesamiento. $a_i = r_i$, $b_i = d_i$ con $b_i - a_i > t_i$. Este caso es equivalente al hecho de considerar ventanas temporales para el comienzo de los trabajos.

Tanto en el caso *a)* como en el *b)*, la atención de los investigadores se ha centrado fundamentalmente en sistemas con máquinas en paralelo. A continuación se señalan los trabajos más relevantes.

La programación de máquinas en paralelo incluye un amplio número de problemas diferentes. [Cheng, 1990] realiza un estado del arte sobre la investigación llevada a cabo dentro de este campo hasta ese instante, aunque pasa por alto los entornos para los que los trabajos poseen un intervalo fijo para el procesamiento, es decir, en todos los problemas que presenta, se supone un intervalo infinito (el trabajo puede ejecutarse en cualquier instante). Es obvio que la limitación del procesamiento de los trabajos a un intervalo no ha tenido gran relevancia dentro de la programación de trabajos en máquinas. En la mayoría de los problemas de secuenciación, incluso cuando el criterio está relacionado con las fechas de entrega, los trabajos no poseen limitaciones de intervalo de proceso, y por ello las soluciones conllevan el procesamiento de todos los trabajos.

Los problemas que se ocupan de esta limitación en entornos con máquinas en paralelo son los siguientes:

- Cuando el intervalo de procesamiento de cada trabajo coincide con su tiempo de proceso, el problema se denomina Programación de trabajos fijos (*Fixed job scheduling problem, FSP*).

- Cuando el intervalo de procesamiento es mayor que el tiempo de proceso del trabajo, el problema se denomina programación de trabajos variables (*Variable job scheduling problem, VSP*).

3.4.1.1. Programación de trabajos fijos (FSP)

En el problema FSP se disponen de n trabajos $N=\{1...i...n\}$, con fecha de inicio y tiempo de procesado dados. Se define para un trabajo J_i, s_i como el instante de comienzo, f_i el instante de finalización y $t_i = s_i - f_i$ la duración del mismo. Se asocia también con cada trabajo un valor w_i o beneficio asociado a la realización del mismo. Los trabajos deben ser procesados sobre un conjunto de m máquinas paralelas. *FSP* se reduce al problema de determinar si es posible programar todos los trabajos, teniendo en cuenta las siguientes restricciones:

- Cada trabajo se procesa de forma ininterrumpida sobre alguna de las máquinas del sistema.

- Cada máquina no puede procesar más de un trabajo al mismo tiempo.

- Los trabajos y las máquinas están divididos en clases de trabajos y clases de máquinas, respectivamente, de forma que cada clase de máquinas sólo puede procesar un subconjunto de clases de trabajos.

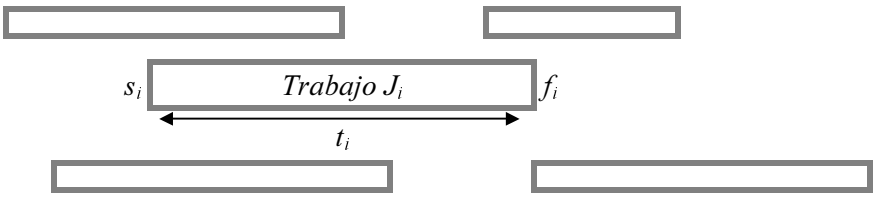


Figura 3.6: Diagrama temporal de los trabajos en el problema FSP

Este problema juega un importante papel en algunos sistemas no productivos, como podrían ser la asignación de conductores a autobuses en una compañía de transportes, la planificación de una agencia de alquiler de casas en zonas turísticas, la asignación de puertas de entrada a vuelos en aeropuertos, etc.

A continuación analizaremos dos enfoques del problema FSP para el caso de una sola clase de máquinas y trabajos. El primer enfoque se centra en el cálculo del número mínimo de máquinas necesarias para procesar todos los trabajos. El segundo enfoque parte de un número fijo de máquinas, insuficientes para el proceso de todos los trabajos, y el problema se centra en determinar qué trabajos serán procesados. Este segundo enfoque se ha denominado Max. FSP.

Finalmente, en la sección 3.4.1.1.3 se analiza brevemente el problema con varias clases de trabajos y máquinas.

3.4.1.1.1. Cálculo del número mínimo de máquinas

En este enfoque, la resolución del problema FSP se limita al cálculo del grado de simultaneidad de los trabajos en el tiempo, según el siguiente lema propuesto por [Kroon et al, 1995]:

Lema: *Una programación admisible para todos los trabajos existe si y sólo si el máximo grado de simultaneidad de los trabajos es menor o igual al número de máquinas m.*

Supongamos n trabajos para procesar en el intervalo $[0,T]$. El grado de simultaneidad en cada instante de tiempo se define como:

$$L_t = \{i \in N : s_i \leq t \leq f_i\text{-}1\}$$

El máximo grado de simultaneidad queda definido como:

L = máximo $\{L_t : 0 \leq t \leq T\}$

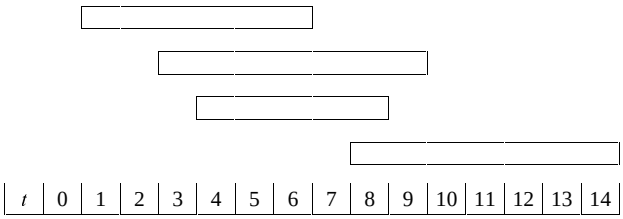


Figura 3.7. Ejemplo del problema FSP con 4 trabajos

La figura 3.7 muestra un ejemplo de un problema con 4 trabajos, representados mediante barras horizontales. El cálculo del grado de simultaneidad sería el siguiente:

t	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
L_t	0	1	1	2	3	3	3	2	3	2	1	1	1	1	1

L = máximo{0, 1, 1, 2, 3, 3, 3, 2, 3, 2, 1, 1, 1, 1, 1} = 3

Es claro que sólo si el número de máquinas es mayor o igual a 3 se podrían realizar todos los trabajos.

Si bien el objetivo descrito para el problema trata únicamente de averiguar si es posible procesar todos los trabajos, también ha sido estudiado en la literatura el caso en el que el número de máquinas es ilimitado y el objetivo es el de minimizar el número de máquinas necesarias para procesar todos los trabajos [*Gertsbakh* y Stern, 1977] [Fischetti et al, 1990]. En este caso el problema se define de la siguiente forma: Sean n trabajos con tiempos dados de comienzo s_i y finalización f_i. Cada trabajo debe ser procesado, sin interrupción, sobre una máquina del conjunto ilimitado de máquinas idénticas disponibles. Una máquina puede procesar cualquier trabajo, pero no más de un trabajo al mismo tiempo. El objetivo del problema es encontrar el menor número de máquinas necesarias para procesar todos los trabajos.

Es obvio que la resolución de este problema equivale simplemente al cálculo del mayor grado de simultaneidad, realizado anteriormente. Sin embargo, el problema puede resolverse mediante un simple algoritmo de asignación avaricioso que obtiene además la asignación óptima de trabajos a máquinas [Fishetti et al, 1990]. A pesar de asumir un número de máquinas ilimitado, es equivalente considerar un número de máquinas lo suficientemente grande de forma que siempre existan máquinas disponibles. Una cota superior para el número de máquinas podría ser el número de trabajos (cada máquina procesaría un trabajo). Definido el conjunto de trabajos

como $J = \{J_1, ..., J_i, ...,J_n\}$, podemos por ello definir el conjunto de máquinas como $M = \{M_1, ..., M_j, ..., M_m\}$ con $m \geq n$.

La estrategia que sigue el algoritmo consiste en recorrer el conjunto de trabajos, ordenados por fecha de inicio s_i, e ir asignando una máquina libre a cada trabajo. En la asignación se exploran primero las máquinas que ya han realizado algún trabajo y que denominaremos activadas(M_A). De entre ellas, el trabajo es asignado a una cualquiera de las máquinas que esté libre (M_L). Si todas las máquinas activadas estuvieran ocupadas se activaría una nueva máquina para realizar el trabajo. Inicialmente ambos conjuntos M_A *y* M_L están vacíos. El pseudocódigo para el algoritmo, denominado *GREEDY*, se muestra a continuación:

```
Procedimiento GREEDY
        M_A = ∅
        M_L = ∅
        a = 0
        Para cada J_j ∈ J
          *Actualizar_Conjunto_M_L (s_i)
          Si M^L = ∅ entonces
            a = a + 1
            M_A = M_A + {M_a}
            Asigna J_j a M_a
          si no
            Selecciona una Maquina M_i ∈ M^L
            M_L = M_L - {M_i}
            Asigna P_j a M_i
          Fin Si
        Fin Para
Fin
```

* El procedimiento "Actualizar_Conjunto_M_L (s_i)" introduce en M_L aquellas máquinas de M_A que hayan quedado liberadas para el instante s_i. La variable a devuelve el número de máquinas utilizadas.

3.4.1.1.2. Max. FSP

Cuando el número de máquinas es menor que el mayor grado de simultaneidad de los trabajos, el problema FSP llega a ser más interesante. En este caso el objetivo del problema pasa a ser el de encontrar el subconjunto de trabajos a ser procesados. Para la selección de trabajos se atiende al criterio del máximo valor total del conjunto de trabajos seleccionados. Cuando suponemos el mismo valor w_i para todos los

trabajos, el objetivo se convierte en maximizar el número de trabajos procesados. Encontramos por tanto un problema de secuenciación en el que no se realizan todos los trabajos, sino solamente un subconjunto de ellos, elegido en base a los pesos de los mismos.

Max. FSP ha sido considerado, entre otros, por [Arkin y Silverberg, 1985], [Kolen et al, 1987] y [Gabrel, 1995], quienes muestran que puede ser resuelto en tiempo polinomial por un algoritmo de flujo a coste mínimo. [Kroon et al, 1995] proponen para la resolución del problema la aplicación de un algoritmo de flujo de coste mínimo a un grafo dirigido $G(N,A)$ cuyo construcción es más directa que la propuesta por los autores anteriores. Para la formación del grafo, se define un conjunto de instantes temporales $\{t_r \ / \ r = t_0...R\}$, ordenado cronológicamente y formado por los instantes de comienzo y finalización de los n trabajos. Es decir, $\{t_r \ / \ r = t_0...R\} = \{s_i, f_i \ / \ i = 1...n\}$. El conjunto $\{t_r, r = t_0...R\}$ constituye el conjunto N de nodos del grafo G. Cada trabajo J_i tiene asociado un arco desde el nodo que representa el instante de comienzo del pedido, s_i, hasta el nodo que representa el instante de finalización f_i con coste igual a $-w_i$ y una unidad de capacidad. Además, existe un arco de coste cero y capacidad ilimitada desde cada nodo t_r a t_{r+1}, $r = 1...R\text{-}1$.

En el grafo se inyectan m unidades de flujo en el primer nodo que serán absorbidas por el último nodo del grafo. Un trabajo será procesado si en la solución óptima del problema, una unidad de flujo circula por su correspondiente arco.

En la figura 3.8 se representa un ejemplo de la construcción del grafo G para 6 trabajos. El grafo resultante se muestra en la figura 3.9.

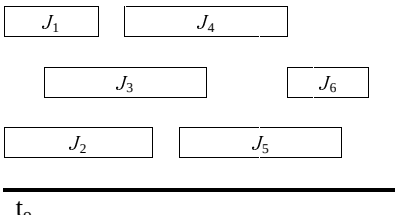


Figura 3.8: Ejemplo Max. FSP

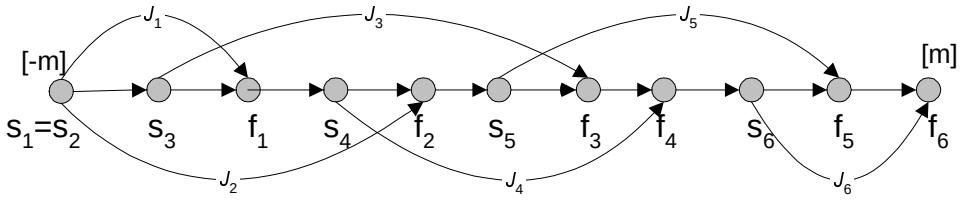


Figura 3.9: Grafo G para el ejemplo de figura 3.8

[Kroon et al, 1995] introducen además un procedimiento de reducción del grafo descrito en el siguiente algoritmo:

Paso 1. Buscar un par de nodos (*i*-1, *i*) en *G* tal que de *i*-1 no parta ningún arco o que al nodo *i* no llegue ningún arco de entre los arcos que representan trabajos. Si no se encuentra ningún par entonces FIN. En otro caso ir al paso 2.

Paso 2. Reemplazar el par de nodos (*i-1*, *i*) por un solo nodo al que lleguen y salgan los arcos que lo hacían en el nodo *i-1* e *i*. Volver al paso 1.

En la figura 3.10 se presenta el grafo reducido sobre el grafo de la figura 3.9.

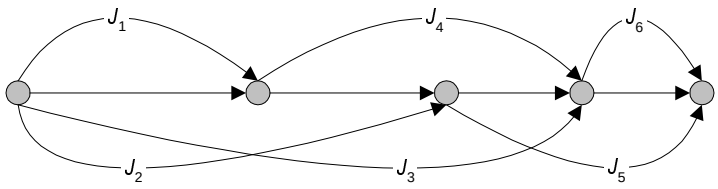


Figura 3.10: Grafo reducido de figura 3.9

3.4.1.1.3. FSP con varias clases de máquinas y trabajos

Cuando hablamos de varias clases de máquinas y trabajos nos enfrentamos al caso más general y complejo del problema FSP. Asumimos que existen *A* diferentes clases de trabajos y *B* diferentes clases de máquinas. Cada trabajo viene determinado por los siguientes atributos: un tiempo de comienzo s_i un tiempo de finalización f_i un valor w_i y una cierta clase de trabajo a_i a la que pertenece.

Las *m* máquinas están dispuestas en *B* clases diferentes, en donde m_b representa el número de máquinas disponibles de la clase *b*.

La admisibilidad entre clases de trabajos y clases de máquinas se representa en una matriz *L* binaria de dimensión *A*×*B*. La interpretación de los elementos de la matriz sería la siguiente:

L_{ab} = 1 si está permitido asignar trabajos de la clase *a* a máquinas de la clase *b*. Ejemplos de matrices podrían ser los siguientes:

L = (1) Sería el caso particular estudiado en la sección anterior. Una misma clase compatible de máquinas y de trabajos.

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

Esta matriz representa una situación con 3 clases de trabajos y 2 clases de máquinas en donde la clase *b* = 1 de máquinas pueden procesar trabajos de la clase *a* = 1 y *a* = 3, y la clase de máquinas *b* = 2 puede

procesar trabajos de la clase $a = 2$ y $a = 3$.

Para este problema el objetivo vuelve a ser de nuevo encontrar el subconjunto de trabajos que pueden ser procesados con máximo valor total.

En el caso de tener una sola clase de máquinas, estaríamos ante el caso particular estudiado en la sección anterior donde vimos que puede ser resuelto en tiempo polinomial; cuando las clases de máquinas son 2 el problema es NP-duro [Kolen, 1992]; si el número de clases de máquinas que existen es de al menos 3 el problema es NP-Completo [Kolen, 1992], excepto para casos triviales. [Arking y Silverberg, 1987] presentan para estos casos un algoritmo basado en Programación Dinámica que puede ser usado para resolver el problema en un tiempo de orden $O(n^{m+1})$. Lógicamente, la funcionalidad del método es pequeña para problemas grandes. Este procedimiento será de nuevo mencionado como idea base para la construcción de métodos exactos en los escenarios del problema PDP.

El problema también es considerado por [Brucker y Norman, 1994] y [Faigle et al, 1998] a través del problema de asignación de *k*-vías (*k-track assignment problem*).

3.4.1.2. Programación de trabajos variables (VSP)

El problema de la planificación de trabajos con tiempos de comienzo variables (*Variable Job Scheduling Problem, VSP*) plantea la siguiente situación: Dados n trabajos con tiempos de proceso t_i fijos y un intervalo de comienzo $[a_i,b_i]$. Cada trabajo debe ser procesado sin interrupción sobre alguna de las máquinas disponibles. Una máquina no puede procesar más de un trabajo al mismo tiempo. El objetivo es encontrar el tiempo de comienzo de cada trabajo tal que el número de máquinas requeridas para procesar todos los trabajos sea mínimo. El problema FSP analizado en la sección anterior puede considerarse como un caso particular del VSP en el que el intervalo corresponde a un tiempo fijo, a un instante de tiempo.

Al igual que ocurría con el problema FSP, de este problema surgen varias versiones atendiendo al número de vehículos disponibles o a la clase de trabajos y máquinas.

A continuación, presentamos la formulación discreta propuesta por [*Gertsbakh y Stern*, 1983] que servirá como referencia para el modelado del problema PDP con ventanas temporales. Dichos autores dividen el tiempo en un conjunto de T intervalos indexados por el índice $j = 0...T\text{-}1$, con el j-ésimo intervalo representando las unidades de tiempo entre $[j,j+1)$. Para cada trabajo i con ventana temporal de comienzo $[a_i,b_i]$ definen el conjunto $T_i = \{a_i,....,d_i\text{-}1\}$ con $d_i = b_i+t_i$ como el conjunto total de instantes dentro de los cuales debe procesarse el trabajo i. Se define también G_j como el conjunto de trabajos asignables al j-ésimo instante de tiempo.

Sea M el número mínimo de máquinas necesarias para procesar los trabajos y sean x_{ij} ($i = 1...n; j \in T_i$) variables binarias, que toman valor 1 si el trabajo i se procesa en el instante j y 0 en otro caso. La formulación mediante programación entera del problema VSP quedaría como sigue:

$$\text{Minimizar } M$$

$$\text{sujeto a}$$

$$\sum_{j \in T_i} x_{ij} = t_i \quad i \in N \quad (1)$$

$$\sum_{i \in G_j} x_{ij} \leq M \quad j = 0, \ldots, T-1 \quad (2)$$

$$t_i x_{ij} - t_i x_{i,j+1} + \sum_{k=j+2} x_{ik} \leq t_i \quad i \in N, \; j \in T_i \quad (3)$$

$$x_{ij} \in \{0,1\} \quad M \geq 0 \text{ Entera}$$

Las restricciones (1) aseguran que cada trabajo es asignado exactamente al número de instantes de tiempo a que corresponde su tiempo de proceso. Las restricciones (2) controlan que no más de M trabajos sean asignados a cada período. Las restricciones (3) aseguran que cada trabajo es asignado a un conjunto adyacente de períodos.

Más complejo aún resulta ser el problema VSP cuando se desea obtener el subconjunto de trabajos a procesar para un número fijo de máquinas en paralelo, y al que podemos denominar *Max VSP*. *Max VSP* es NP-completo cualquiera que sea el número de máquinas disponibles [*Garey y Johnson*, 1979]. [Gabrel ,1995] propone un método de resolución basado en un procedimiento para el problema FSP con varias clases de trabajos. [Chuzhoy y Ostrovsky, 2001] estudian el problema VSP para el caso de una sola máquina proponiendo una aproximación algorítmica de carácter *greedy*. Los resultados de su estudio pueden enfocarse a la resolución del problema para el caso de múltiples plantas. [Krammer y Lee, 2002] resuelven el problema VSP para el caso de 2 máquinas mediante un algoritmo de programación dinámica.

3.4.2. Secuenciación de trabajos en sistemas *just-in-time*

En los entornos *just-in-time*, los trabajos completados con prontitud deben ser depositados en inventario hasta su fecha de entrega, mientras que los trabajos que se completan con retraso provocan una disminución en la calidad del servicio. Por tanto, una programación ideal sería aquella en la que todos los trabajos finalizan justo en su fecha de entrega.

En la mayor parte de la literatura sobre secuenciación de trabajos, en la que se han tratado medidas de comportamiento respecto a fechas de entrega, los criterios han

ido fundamentalmente referidos a medir el retraso (ΣL_i,$\Sigma L_i/n$, $\Sigma w_i L_i$, L_{max}), la tardanza(ΣT_i, $\Sigma T_i/n$, $\Sigma w_i T_i$, T_{max}) o los trabajos retrasados(ΣX^T_i, $\Sigma w_i X^T_i$) [Lawer et al, 1993]. Tanto en los objetivos referidos a tardanza como a trabajos retrasados, la finalización de trabajos con prontitud a su fecha de entrega no supone ninguna variación en la función objetivo. Estos objetivos se aplican en sistemas en los que la prontitud no supone ningún coste.

Sin embargo, durante las últimas décadas el interés en problemas en los que se tiene en cuenta tanto la prontitud como la tardanza ha ido creciendo, estimulado sobre todo por nuevas tendencias en la dirección de los sistemas, entre las que destaca el *just-in-time*. En los sistemas *just-in-time* se valora tanto el retraso como la prontitud de los trabajos, lo que se traslada a la función objetivo de diversas formas. El objetivo más común y más ampliamente utilizado ha sido el de medir la desviación de los instantes de finalización de los trabajos respecto a las fechas de entrega($\Sigma(E_i + T_i)$ o el caso general $\Sigma(u_i E_i + v_i T_i)$). Sin embargo, también se ha hecho referencia al caso en el que únicamente se valoran los trabajos finalizados sin retraso ni adelanto, es decir, $Max\ \Sigma w_i X_i$. A continuación se detallan los trabajos existentes para esos dos objetivos.

3.4.2.1. Maximizar el peso de los trabajos completados en su fecha de entrega

El objetivo sería pues $Max\ \Sigma w_i X_i$. Este criterio puede particularizarse, para el caso de máquinas en paralelo, en el problema Max. FSP comentado en la sección anterior, cuando el instante de llegada del trabajo al sistema es $r_i = d_i - t_i$. Además, basta con pensar que en el problema Max. FSP el conjunto de trabajos no seleccionados es procesado tras la finalización de todos los trabajos. En general, la mayoría de los modelos *just-in-time* asumen que los trabajos están todos disponibles al comienzo del periodo ($r_i = 0$), lo cual supone una cierta desviación respecto a modelos reales.

Centrándonos en el estudio del problema $Max\ \Sigma w_i X_i$ sobre máquinas en paralelo, [Hiraishi et al, 2002], asumiendo tiempos de *set-up* entre trabajos, proponen un método de resolución basado en el cálculo de flujo a coste mínimo en un grafo. La idea para la construcción de este grafo puede ser utilizada para la resolución del problema Max. FSP.

[Lann y Mosheiov, 2002] proponen un algoritmo de orden $O(n \log n)$ para el caso en el que todos los trabajos poseen el mismo peso, y el objetivo se centra en maximizar el número de trabajos finalizados en su fecha de entrega.

Debido a un posterior uso del grafo propuesto en [Hiraishi et al, 2002], presentamos a continuación el modelo de construcción.

- **Grafo propuesto por [Hiraishi et al, 2002]**

Por cada trabajo J_i existen nodos s_i y f_i que representan el comienzo y fin de su procesamiento. Estos nodos están unidos por un arco con coste igual a w_i (peso del trabajo). Existe un nodo inicial s conectado a cada nodo s_i del grafo y un nodo final t al que se conectan todos los nodos f_i del grafo. El coste de estos arcos es nulo. Finalmente, se conectan con coste nulo, nodos f_i con nodos s_j en los casos en que $d_j \geq d_i + s_{ij} + t_j$, donde s_{ij} define el coste de *set-up* entre los trabajos i y j.

Esta idea puede servir para el problema FSP asumiendo costes de *set-up* nulos. En la figura 3.12 se muestra el grafo construido sobre el ejemplo de figura 3.11.

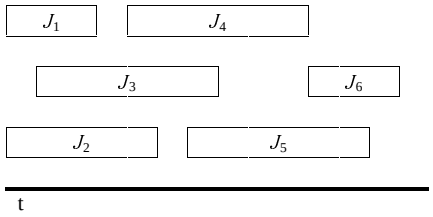


Figura 3.11: Ejemplo de problema FSP

En el grafo se inyectan m unidades de flujo en el nodo s, donde m representa el número de máquinas disponibles en el sistema. Estas m unidades de flujo son absorbidas por el nodo final t. Supuestos los valores de los trabajos que indica la tabla 3.1, la figura 3.13 muestra la correspondiente solución óptima de flujo a coste máximo. Se ha calculado la solución de coste máximo porque se han considerado pesos positivos en el grafo. El número de máquinas consideradas en el ejemplo fue de 2. La solución procesa los trabajos J_1, J_2 , J_4, J_5 y J_6.

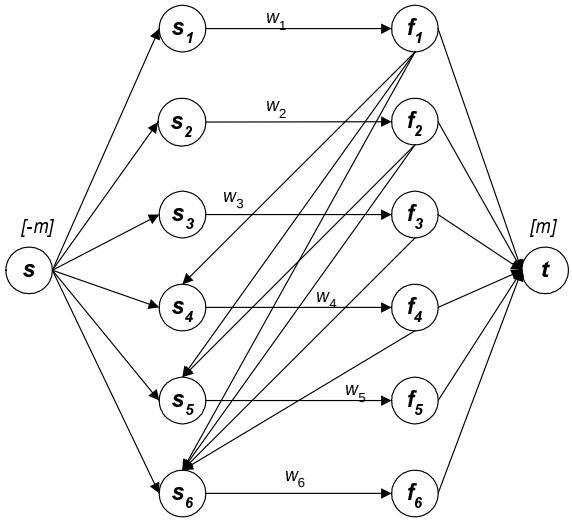


Figura 3.12 : Grafo propuesto en [Hiraishi et al, 2002]

w_1	W_2	w_3	w_4	w_5	w_6
2	2	3	4	4	2

Tabla 3.1: Tabla de pesos para el problema de la Figura 3.11

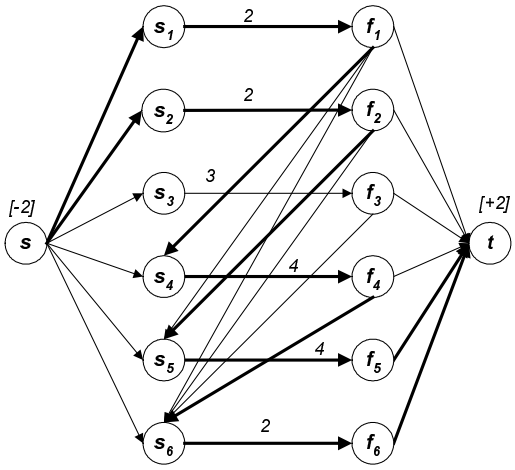


Figura 3.13 : Solución de flujo a coste máximo para la Figura 3.12

El número de nodos en el grafo es siempre $2n+2$. El número de arcos varía en función de los tiempos de los trabajos. Siempre existen $3n$ arcos fijos más los que se generen entre trabajos no solapados, es decir, aquellos en los que $d_j >= d_i + t_j$.

La siguiente tabla muestra una comparativa del orden en el número de nodos y arcos de los grafos de [Kroon et al, 1995] e [Hiraishi et al, 2002].

	Grafo [Kroon et al, 1995]	Grafo [Hiraishi et al, 2002]
Número de Nodos	2n	2n + 2
Número de Arcos	2n + 1	≥ 3n ≤ 3n + (n^2 – n)/2

Tabla 3.2: Comparación de estrategias

De la tabla 3.2 se desprende la conveniencia del uso del grafo de Kroon et al, si bien esta diferencia no es del todo importante por la ya mencionada complejidad polinomial del algoritmo para el cálculo del flujo a coste mínimo en un grafo. En el estudio del primer escenario del problema PDP se describirá una nueva construcción con menor número de nodos y arcos que las dos implementaciones mencionadas.

3.4.2.2. Penalización de retrasos y adelantos($\Sigma(u_iE_i + v_iT_i)$)

Los problemas de secuenciación que minimizan el valor total de los retrasos y adelantos de los trabajos (*weighted earliness and tardiness*) se presentan en sistemas de producción *just-in-time* donde el objetivo es completar los trabajos lo más cercano posible a su fecha de entrega. Los problemas de secuenciación con

penalizaciones para adelantos y retrasos se muestran como los más comunes dentro de los sistemas *just-in-time*.

[Baker y Scudder, 1990] realizan una revisión de esta clase de problemas, denominados problemas E/T, o simplemente ET. Como se detalla en dicha revisión, la mayoría de los modelos que incorporan penalizaciones de retraso y adelanto son modelos de una sola máquina que involucran una fecha común de entrega para todos los trabajos. Para múltiples máquinas, únicamente se mencionan los trabajos realizados por [Hall, 1986], [Sundararaghavan y Ahmed, 1984] y [Emmons, 1986], todos ellos con fecha común de entrega para todos los trabajos. [Zhu y Heady, 2000] realizan una breve revisión de los problemas ET sobre múltiples máquinas, sin hacer alusión a problemas con diferentes fechas de entrega para los trabajos. Tanto para el caso de una sola máquina como máquinas en paralelo y asumiendo la misma fecha de entrega para todos los trabajos, el problema es NP-duro [Chen y Powell, 1999], lo cual viene a ser un caso particular del problema general con diferentes fechas de entrega.

[Sivrikaya-Serifoglu y Ulusoy, 1999] estudian el problema con fechas de llegada al sistema y fechas de entrega diferentes. Sería este el caso más próximo en la literatura en lo referente al problema VSP, con penalización respecto a una fecha ideal de entrega. Sin embargo, incluyen tiempos de *set-up* dependientes de la secuencia y dos tipos de máquinas, idénticas y uniformes.

3.4.3. Sistemas de flujo uniforme con máquinas en paralelo (FSPM)

El problema FSPM fue definido por primera vez por [Arthanari y Ramaurthy, 1971] y posteriormente por [Salvador, 1973]. Dentro de los problemas FSPM también pueden incluirse centros con una sola máquina, pero al menos una de las estaciones debe estar formada por múltiples máquinas. Los más estudiados en la bibliografía son los formados por 2 estaciones de máquinas ([Gupta, 1988] [Chen, 1995], [Haouari y M'Hallah, 1997][Huang y Li, 1998], [Li, 1997]).

Los problemas FSPM han atraído la atención de muchos investigadores en los últimos años, mostrándose en muchos sistemas reales de fabricación ([Paul, 1979], [Pinedo, 1995], [Ramudhin y Ratliff, 1995] [Drees y Wilhelm, 2001] y [Riane et al, 2001]). El objetivo perseguido en la amplia mayoría de ellos es la minimización del *makespan* o tiempo de finalización del último trabajo [Linn y Zhang,1999]. Respecto a objetivos relacionados con fechas de entrega, [Paul, 1979] minimiza el número de trabajos retrasados y [Adler, 1993] considera como objetivo la suma de los instantes de retraso. [Ramudhin y Ratliff, 1992] estudian el problema FSPM con dos centros y considerando como objetivo maximizar la suma de pesos de los trabajos completados dentro de un cierto horizonte temporal. Supone este problema,

por tanto, un caso en el que no son realizados todos los trabajos. Sin embargo, se supone una misma fecha tope de entrega para todos los trabajos.

Las técnicas exactas de optimización utilizadas hasta ahora para la resolución del problema FSPM incluyen fundamentalmente métodos de exploración dirigida. Como ejemplo, [Brah y Hunsucker, 1991] desarrollan una exploración dirigida a partir de la idea propuesta por [Barkley et al, 1975] para un problema con máquinas en paralelo.

A causa del carácter NP-Completo de esta clase de problemas, los procedimientos aproximados de resolución han sido los más utilizados para la resolución de este problema [Chen,1995] [Haouari y M'Hallah, 1997] [Nowicki y Smutnicki, 1998].

- **Sistemas FSPM con *no-wait* en proceso**

En muchos entornos de fabricación, existe la restricción de que una vez que un trabajo ha comenzado su procesamiento, debe ser procesado por completo sin ninguna interrupción entre las máquinas. Tales sistemas llevan el adjetivo de *no-wait* en proceso(no espera en proceso). Este tipo de secuenciación es necesaria cuando las operaciones sobre los trabajos requieren un proceso continuo sin interrupciones, debido a las restricciones de producción. Como ejemplos, podemos destacar la producción de molduras de plástico, la colada continua de acero [Eguia, 1996], producción de bloques de hormigón [Grabowski y Pempera, 2000] y determinados procesos químicos.

No-wait en proceso ha sido una de las características utilizadas en los problemas FSPM. [Hall y Sriskandarajah, 1996] realizan una completa revisión de problemas de secuenciación con *no-wait* en proceso. En dicha revisión destacan algunos problemas FSPM, la mayor parte de ellos referidos al caso de 2 estaciones de máquinas con alguna de las estaciones formada por una sola máquina. Para el caso de varias máquinas en cada estación, destaca el trabajo de [Ramudhin y Ratliff, 1992] anteriormente referido. El resto de aportaciones van referidas a la minimización del *makespan*.

3.5. TÉCNICAS DE RESOLUCIÓN

En general, las técnicas para la resolución de problemas combinatorios involucran modelos y métodos de elección discreta que tienen como raíz la teoría de la programación lineal y están fuertemente vinculados con la matemática discreta, la computación algorítmica y la teoría de la complejidad.

Un punto importante en la teoría de la secuenciación de trabajos en máquinas es la aparición de la teoría de la complejidad algorítmica [Garey y Jonson, 1979]. A través de dicha teoría, podemos demostrar que algunos de los problemas de secuenciación son demasiado difíciles de resolver de forma óptima. Esto significa que es poco o nada probable que exista un algoritmo que en tiempo polinomial, es decir, en un número de pasos de orden polinomial, resuelva el problema. Hasta la actualidad, sólo unos pocos problemas del campo de la secuenciación se distinguen por estar fuera de este ámbito, es decir, por ser fácilmente tratables.

Desde la aparición de la teoría de la complejidad, la tendencia en la investigación ha sido la de encontrar algoritmos polinomiales exactos cuando el problema es resoluble polinomialmente. Cuando no es posible esto último, la atención se ha centrado en el diseño de algoritmos heurísticos que sean rápidos y que produzcan soluciones “próximas” a la solución óptima. Para este último caso es también interesante el diseño de algoritmos exponenciales exactos basado en la enumeración de soluciones, con objeto de resolver instancias de un tamaño “adecuado”. Si el tiempo de ejecución crece de modo exponencial con el tamaño de la instancia, no podrán resolverse instancias de cualquier tamaño, pero al menos sí se podrán resolver instancias “pequeñas”. Además, la definición de la complejidad de un problema se hace sobre el peor caso, es decir, la complejidad es el número máximo de pasos a realizar. Sin embargo, puede ocurrir que instancias concretas de un cierto tamaño puedan ser resueltas en un tiempo razonable.

Para la resolución del problema PDP se han abordado los tres tipos de algoritmos definidos con anterioridad:

- Algoritmos polinomiales que alcanzan el óptimo del problema.
- Técnicas de resolución exacta basadas en la teoría de grafos.
- Técnicas heurísticas

A continuación se describen las técnicas de resolución empleadas en la resolución del problema en sus diferentes escenarios.

3.5.1. Técnicas de resolución exacta basadas en la teoría de grafos

Si bien el objetivo primordial del trabajo es la investigación sobre técnicas heurísticas para la resolución del problema, también ha sido necesario la implementación de procedimientos exactos que proporcionen los valores óptimos de los problemas, con objeto de poder evaluar la validez de los métodos heurísticos implementados. Debido a la dificultad del problema en la mayoría de los escenarios, los métodos exactos implementados no han sido eficientes ni en tiempo

ni en consumo de memoria para instancias de tamaño medio y grande, lo que ha repercutido en el tamaño de los problemas resueltos.

Mucha de la investigación sobre el diseño de algoritmos rápidos para problemas de flujos se inicia a partir del trabajo de [Tardos, 1985], quien encontró el primer algoritmo polinomial puro para el problema de flujo a coste mínimo. Para un grafo con n nodos y m arcos, son conocidos algoritmos polinomiales para el problema de flujo a coste mínimo con tiempos de ejecución de orden $O(m \log n (m + n \log n))$ [Ahuja et al, 1993].

Gracias a las mejoras alcanzadas en los algoritmos de resolución de problemas asociados a grafos, como el cálculo de flujo máximo o el flujo a coste mínimo, muchos problemas de ingeniería de organización han sido implementados como problemas sobre grafos. La variedad de aplicaciones en el campo de la optimización en grafos no sólo se presenta en problemas asociados a sistemas de redes físicas, sino también en situaciones que aparentemente no presentan ninguna relación con una red. Uno de los ejemplos más característicos se produce en los problemas de planificación de la producción, donde frecuentemente se modelan situaciones resueltas con algoritmos de ruta mínima y de flujo a coste mínimo en un grafo.

El modelo de flujo a coste mínimo es uno de los problemas fundamentales en la teoría de grafos. Dado un grafo en el que existen determinados nodos que demandan mercancías de otros nodos que la suministran, en este problema se desea determinar el coste menor de envío de esas mercancías a través del grafo, suponiendo un coste en cada uno de los arcos. Este modelo posee un amplio número de aplicaciones en casi todos los campos, destacando su aplicación en los problemas de distribución [Glover y Klingman, 1976]. En el campo de la programación de tareas y asignación de recursos también son muchos los modelos basados en grafos, algunos de ellos han sido ya referidos en el capítulo anterior. Otros problemas relacionados son:

- Problema de asignación de terminales [Esau y Williams, 1966]
- Programación en sistemas just-in-time [Elmaghraby, 1978]; [Levner y Nemirovsky, 1991]
- Almacenamiento y distribución [Jewel, 1957]

Existen diversos métodos de cálculo para la resolución del problema de flujo a coste mínimo [Ahuja et al, 1993]. En esta investigación se ha utilizado la implementación en C++ de [Frangioni, 2001] del algoritmo RELAX IV [Bertsekas y Tseng, 1988].

Otro problema utilizado en esta tesis ha sido el cálculo de la ruta mínima en un grafo dirigido. En este problema, se desea encontrar el camino desde un nodo origen

a un nodo destino asumiendo que cada arco posee un coste asociado. De forma análoga, se define el problema de la ruta máxima en un grafo. Este problema puede ser considerado un caso especial del problema de flujo a coste mínimo, con el nodo de inicio suministrando una unidad de flujo que demanda el nodo final del grafo. Los grafos con los que nos enfrentamos a lo largo de todo el estudio, además de ser dirigidos, no contienen ciclos y son directamente enumerados, lo que facilita la obtención del camino mínimo. El algoritmo que se ha utilizado para el cálculo del camino máximo en este tipo de grafos aparece descrito en [Larrañeta, 1999].

En las técnicas de resolución exacta empleadas en la tesis, la complejidad del método de solución no depende de los algoritmos de resolución utilizados, puesto que, como se ha comentado, éstos presentan una complejidad polinomial respecto al tamaño del grafo. La complejidad se encuentra en el tamaño del grafo. Cuando nos hemos referido a algoritmos polinomiales que resuelven el problema PDP en algún escenario concreto, estamos diciendo que el grafo construido para la resolución presenta un tamaño polinomial respecto al tamaño del problema. Sin embargo, en la mayoría de los métodos exactos implementados, el tamaño del grafo alcanza un número de nodos y arcos de orden exponencial, lo que dificulta la resolución exacta del problema. En estos casos, y como se expondrá detalladamente en cada escenario, los grafos utilizados recogen soluciones admisibles de la planificación de los pedidos, de una forma constructiva.

3.5.2. Técnicas heurísticas

Dada la dificultad práctica para resolver de forma exacta importantes problemas combinatorios, surgen las técnicas heurísticas como algoritmos que proporcionan soluciones factibles, las cuales se supone que al menos se acercan al valor óptimo en un tiempo de cálculo razonable. No se trata de soluciones óptimas, pero sí de soluciones que pueden calificarse como satisfactorias.

Una importante ventaja que ofrecen estas técnicas respecto a las que buscan soluciones exactas es que, por lo general, permiten una mayor flexibilidad en el manejo de las características del problema.

También presentan inconvenientes, uno de los más destacados es que por lo general no es posible conocer la calidad de la solución sin conocer el valor del óptimo del problema. Para solventar este problema existen métodos que realizan acotaciones, que pueden dar una orientación respecto a la calidad de la solución obtenida.

[Díaz et al, 1996] clasifican los métodos heurísticas en los siguientes tipos:

- Métodos constructivos. Consisten en ir paulatinamente añadiendo componentes individuales a la solución hasta que se obtiene una solución factible.

- Métodos de descomposición. Se trata de dividir el problema en subproblemas más pequeños, siendo la salida de uno la entrada de su siguiente, de forma que al resolverlos todos se obtiene una solución del problema global.

- Métodos de reducción. Tratan de identificar alguna característica que presumiblemente deba poseer la solución óptima y de ese modo simplificar el problema.

- Manipulación del modelo. Modifican la estructura del modelo con el fin de hacerlo más sencillo de resolver.

- Métodos de búsqueda por entornos. Dentro de esta categoría se encuadran la mayoría de las técnicas conocidas como metaheurísticas. El procedimiento básico de estas técnicas es el siguiente: Parten de una solución factible inicial y mediante alteraciones de esta solución, van pasando de forma iterativa, y mientras no se cumpla un determinado criterio de parada, a otras soluciones factibles de su entorno, almacenando como óptima la mejor de las soluciones visitadas.

Las metaheurísticas más utilizadas y que más éxito han tenido en la resolución de problemas combinatorios han sido:

- Búsqueda Tabú.

- Algoritmos genéticos.

- GRASP.

Para la resolución del problema PDP se han empleado las tres técnicas metaheurísticas anteriormente señaladas. También se han empleado procedimientos heurísticos basados en la manipulación del modelo. A continuación se describe el método genérico de las tres metaheurísticas utilizadas.

3.5.2.1. GRASP

GRASP (*Greedy Randomised Adaptative Search Procedure)* es una metaheurística multicomienzo para optimización combinatoria [Aiex et al, 2000][Resende y Ribeiro, 2003][Pitsoulis y Resende, 2002]. Está basada en un procedimiento de búsqueda avaricioso, aleatorio y adaptativo. Este método, aplicado a una amplia gama de problemas de optimización, garantiza en la mayoría de los casos, una buena solución, aunque suele ser la más débil de entre las técnicas más utilizadas. Las implementaciones GRASP generalmente son robustas, en el sentido de que es difícil encontrar ejemplos patológicos en donde el método funcione arbitrariamente mal.

Los métodos GRASP fueron desarrollados al final de la década de los ochenta con el objetivo inicial de resolver problemas de cubrimientos de conjuntos [Feo y Resende, 1989].

GRASP es un procedimiento iterativo donde cada paso consiste de una fase de construcción y una de mejora. En la fase de construcción se aplica un procedimiento heurístico constructivo para obtener una buena solución inicial. Esta solución inicial se mejora en la segunda fase mediante un algoritmo de búsqueda local. La mejor de todas las soluciones examinadas se guarda como resultado final.

En la fase de construcción se construye iterativamente una solución, considerando un elemento en cada paso. En cada iteración la elección del próximo elemento a añadir a la solución parcial viene determinada por una función *greedy*. Esta función, de carácter avaricioso, mide el beneficio de añadir cada uno de los elementos según la función objetivo y elegir la mejor. Hay que hacer notar que esta medida es miope, puesto que no tiene en cuenta qué ocurrirá en iteraciones sucesivas al realizar una elección, sino únicamente en esta iteración.

El procedimiento es adaptativo porque el valor de la función asociada con cada elemento se modifica en cada iteración de la fase de construcción, para reflejar los cambios producidos por la incorporación a la solución parcial de los elementos previos. Es decir, la valoración que se tenga de añadir un determinado elemento a la solución en una iteración de la fase de construcción, no coincidirá necesariamente con la que se tenga en la siguiente.

El procedimiento es aleatorio porque no selecciona el mejor candidato según la función avariciosa en cada paso, sino que con objeto de diversificar y no repetir soluciones en dos construcciones diferentes, se construye una lista con los mejores candidatos y se toma uno de ellos al azar. Esta lista de los mejores candidatos se denomina *lista restringida de candidatos* (LRC). La lista LRC puede venir definida con un tamaño fijo o con un tamaño variable en función de un parámetro α. Este parámetro determina el grado de aceptación de los candidatos en la lista LRC. Los

candidatos con función *greedy* mayor que un cierto porcentaje del valor del mejor candidato, pertenecen a la lista LRC. Este porcentaje lo define el valor de α. Veamos, como sería el proceso de construcción de esta lista:

LRC ≡ Lista de Candidatos Restrictiva
α = Parámetro de restricción
yj = Función *Greedy*
$y^* = MAX \{y_j\}$
Para j = 1 hasta n
  Si $y_j \geq \alpha\, y^*$ entonces RCL = RCL $\cup$ $\{j\}$
Fin Para

Dado que la fase inicial no garantiza optimalidad local respecto a una estructura de vecindad, se aplica un procedimiento de búsqueda local para mejorar la solución obtenida. En la búsqueda local se suele emplear un procedimiento de intercambio simple con objeto de no emplear mucho tiempo en esta mejora. El algoritmo de búsqueda local actúa de una forma iterativa reemplazando la solución actual por la mejor solución de la vecindad en cada paso. Este proceso termina cuando no se encuentran soluciones mejores a la dada en la vecindad.

Por tanto, para definir un GRASP es necesario establecer los siguientes elementos:

- Una función avariciosa.
- Una estrategia de construcción de soluciones.
- Una técnica de búsqueda local.

Un pseudocódigo del GRASP genérico puede ser el siguiente [Resende, 1998]:

Procedimiento GRASP (MaxIter)
  MejorSolución = 0
  Para *k* = 1…*MaxIter*
    P^S = *ConstruirSolución ()*
    P^L = *BúsquedaLocal* (P^S)
    Si *Valor* (P^L) < *Valor (MejorSolución)* **entonces**
      MejorSolución = P^L
    Fin Si
    Devolver *(MejorSolución)*
  Fin Para
Fin

```
Procedimiento ConstrucciónSolución ()
    Solución = {}
    Mientras [La solución no esté completa]
        Actualizar la función greedy
        Hallar LRC(LRC)
        a = Selección aleatoria de un elemento(LRC)
        Solución = Solución + {a}
    Fin Mientras
Fin
```

GRASP ha sido aplicado a diversos problemas de secuenciación de trabajos. [Laguna y González Velarde, 1991] consideraron la programación de máquinas en paralelo en un entorno de producción basado en *just-in-time*. [Feo et al, 1991] desarrollan un GRASP para un problema con una sola máquina.

3.5.2.2. Búsqueda Tabú

Búsqueda Tabú es un procedimiento heurístico propuesto por [Glover, 1989] [Glover, 1990] que explora el espacio de soluciones a través de repetidos movimientos desde una solución a la mejor de sus vecinas. Búsqueda Tabú ha sido exitosamente usada para resolver complejos problemas de optimización de carácter combinatorio, y en particular en el campo de la secuenciación. Un largo número de aplicaciones de búsqueda tabú para problemas de secuenciación puede encontrarse en la literatura (Glover y Laguna, 1997). Respecto al problema FSPM, métodos eficienes basados en búsqueda tabú son propuestos en [Haouari y M'Hallahm 1997], [Lee, 2001], [Wardono y Fathi, 2003] y [Oguz et al, 2003].

En un amplio sentido, la búsqueda tabú es un proceso iterativo que proporciona un mecanismo para escapar de óptimos locales en el proceso de búsqueda del espacio de soluciones. Para ello introduce el concepto de memoria en el algoritmo y, mediante estructuras simples, lo utiliza para dirigir la búsqueda teniendo en cuenta la historia de la misma.

Al contrario que en la búsqueda local (que siempre se mueve al mejor de su entorno y finaliza con la llegada a un óptimo local), la búsqueda tabú permite moverse a una solución del entorno aunque no sea tan buena como la actual, de modo que se pueda escapar de óptimos locales y continuar estratégicamente la búsqueda de soluciones aún mejores. Para evitar que el proceso vuelva a un viejo óptimo local y se cicle, la búsqueda tabú etiqueta los movimientos más recientes

como movimientos tabú, de forma que no es posible repetirlos durante un determinado número de iteraciones.

Se requieren los siguientes elementos para la definición de un algoritmo de búsqueda tabú:

- *Solución inicial*: La búsqueda debe comenzar desde una solución inicial. Ésta podría ser cualquier solución admisible que satisfaga las restricciones del problema.

- *Movimiento*: Un movimiento es un procedimiento específico por el que se genera una solución admisible desde la solución actual. Suele ser un procedimiento simple para el caso de problemas combinatorios.

- *Vecindad*: Dada una solución *S*, la vecindad N(*S*) es el conjunto de todas las posibles soluciones admisibles generadas por la ejecución de un movimiento sobre la solución actual *S*.

- *Lista tabú*: La lista tabú es un mecanismo de memoria para evitar que la búsqueda cicle o quede atrapada en un óptimo local. La lista tabú incluye un cierto número de movimientos, que no son permitidos en la iteración actual. Una vez que un movimiento, que genera una nueva solución, es aceptado, su movimiento inverso se añade a la lista tabú y permanece en ésta un número determinado de iteraciones, denominado duración tabú.

- *Criterio de parada*: En general, la búsqueda finaliza después de un cierto número de iteraciones, después de un tiempo de computación predefinido, o cuando se alcanza un número dado de iteraciones sin mejorar la mejor solución.

- *Criterio de aspiración*: Es una regla que invalida las restricciones tabú. Si un cierto movimiento se considera prohibido por las restricciones tabú pero satisface el criterio de aspiración, entonces este movimiento puede ser considerado admisible. Por ejemplo, si un movimiento tabú alcanza una solución que mejora la función objetivo de la mejor solución encontrada hasta ese momento, entonces ese movimiento es aceptado.

Dado este conjunto de elementos básicos, el esquema de funcionamiento de una búsqueda tabú puede ser descrito como sigue:

Se comienza en una solución inicial y usando un procedimiento para obtener soluciones vecinas, se genera un conjunto de movimientos candidatos y se evalúan sus correspondientes funciones objetivos. Si el mejor movimiento es tabú pero

satisface el criterio de aspiración, se selecciona y pasa a ser la nueva solución actual. En otro caso, el mejor movimiento no-tabú pasa a ser considerado la nueva solución actual. Repetimos este procedimiento hasta que el criterio de parada se satisfaga. La mejor solución encontrada hasta ese momento es la solución obtenida por el algoritmo. El movimiento que es elegido en cada iteración se introduce en la lista tabú para que similares soluciones no sean generadas de nuevo en las siguientes iteraciones. El tamaño de la lista tabú controla la permanencia de un movimiento dentro de dicha lista.

El tamaño de la lista tabú puede ser usado como medio para controlar la estrategia de búsqueda. Una estrategia basada en intensificación enfoca la búsqueda sobre la región actual del espacio de soluciones. Una estrategia de diversificación consigue mover la búsqueda a nuevas áreas no exploradas del espacio de soluciones. Si la lista tabú es pequeña entonces se tiende a intensificar la búsqueda. En cambio, si es grande, la tendencia es a diversificar.

Un pseudocódigo de la búsqueda tabú puede expresarse como sigue:

```
Procedimiento Búsqueda Tabú (criterio de parada, criterio aspiración)
    Mejor solución ={∅}
    Lista tabú = {∅}
    Solución = Generar solución inicial
    Mientras no se cumpla el criterio de parada
        Generar Vecindad (Solución)
        Evaluar función objetivo (Vecindad)
        Mov Aceptado = Elegir Movimiento(Lista tabú, criterio aspiración)
        Actualizar Lista tabú (Lista tabú, Mov Aceptado, Tamaño Lista)
        Solución actual = Calcular Nueva Solución (Mov Aceptado)
        Si Valor (Solución) > Valor (Mejor solución) entonces
            Mejor solución = Solución actual
        Fin Si
    Fin Mientras
Fin
```

3.4.2.3. Algoritmos genéticos

Los Algoritmos Genéticos son “técnicas de búsqueda basadas en la mecánica de la selección natural y la genética” [Goldberg, 1989]. Introducidos por [Holland, 1975], los Algoritmos Genéticos son flexibles y de ámbito general, y pueden ser aplicados a un amplio número de problemas.

En los organismos biológicos, la información hereditaria es pasada a través de los cromosomas que contienen la información de todos esos factores, es decir, los genes. La evolución ocurre en los cromosomas; un ser vivo da vida a otro mediante la decodificación de los cromosomas de sus progenitores, el cruce de los mismos, y la codificación de los nuevos cromosomas formando los descendientes; las mejores características de los progenitores se trasladan a los descendientes, mejorando progresivamente las generaciones.

Basándose en estas características, [Holland, 1975] creó un algoritmo que genera nuevas soluciones a partir de la unión de soluciones progenitoras utilizando operadores similares a los de la reproducción.

Las soluciones de un cierto problema se codifican mediante cromosomas, llamados también individuos, obteniéndose un valor que se denomina bondad(*fitness*). Siempre existe un conjunto de soluciones almacenadas denominado población. Las soluciones se emparejan a través de ciertos mecanismos de reproducción, obteniéndose nuevas soluciones que se incorporan a la población, desplazando a las de menor bondad.

Las tres operaciones antes mencionadas (reproducción, cruce y mutación) son las usadas en este tipo de algoritmos. Un Algoritmo Genético aplica estas operaciones a una población y genera nuevos individuos de forma iterativa. El proceso de reproducción se guía a través de una función que mide el *fitness* del individuo. Generalmente se asimila la función objetivo del problema con el valor del *fitness*. Los individuos que mejor se adapten, los de mayor *fitness*, tendrán una mayor probabilidad de selección y supervivencia.

Los Algoritmos Genéticos trabajan sobre una población de soluciones, generando una nueva población en cada iteración. La generación de nuevas soluciones es aleatoria por naturaleza. Utilizan una elección al azar para guiar la búsqueda, repitiendo, durante un número relativamente largo de iteraciones, operaciones muy simples que dan lugar a un volumen masivo de soluciones, dentro de la búsqueda de las que se consideran buenas soluciones.

La evolución simulada está diseñada para encontrar cada vez mejores individuos mediante una manipulación “ciega” de su contenido. El término “ciega” se refiere a que el proceso no tiene ninguna información acerca del problema que está tratando de resolver, exceptuando el valor del *fitness*. Un Algoritmo Genético puede ser visto como una estructura de control que organiza o dirige un conjunto de transformaciones y operaciones diseñadas para simular estos procesos de evolución.

- **Individuos: Genotipo y Fenotipo**

Existen dos vistas diferentes de un individuo: El genotipo es la representación del individuo sobre la que el algoritmo trabaja, es decir la representación codificada sobre la que se aplican los operadores de evolución. La decodificación del genotipo da lugar al fenotipo. El fenotipo es la representación del individuo adaptada a las restricciones del problema, y sobre la que se calcula su *fitness*.

Existen dos tipos diferentes de codificación en un algoritmo genético:

- *Codificación directa*: Los algoritmos genéticos con una codificación directa no hacen uso de la distinción entre genotipo y fenotipo. Los dos representan la misma codificación del individuo.

- *Codificación indirecta*: La solución representada por el genotipo necesita una interpretación, según el problema, para la obtención del fenotipo del individuo.

Otro punto importante en la representación del individuo es la codificación usada en el algoritmo. Las codificaciones más habituales son la representación del individuo mediante una cadena de datos binarios, decimales o permutación de enteros.

- **La población inicial**

La población inicial de un algoritmo genético puede ser creada de diferentes maneras. Hacer evolucionar una población que ha sido generada aleatoriamente, hasta llegar a tener una bien adaptada, es un buen test para saber cómo está funcionando la implementación del Algoritmo Genético, debido a que las características esenciales y críticas de la solución final deben ser consecuencia de este proceso evolutivo y no de las características de los métodos usados para generar la población inicial.

- **Evaluación del nivel de *fitness***

El *fitness* es la valoración de la solución que proporciona el individuo y generalmente se asemeja a la función objetivo del problema. La función utilizada para asignar los valores del *fitness* debe ser tal, que dé como resultado una buena discriminación.

- **Técnicas de reproducción de la población**

Dos técnicas para la reproducción de la población son usadas generalmente en el campo de los algoritmos genéticos:

- Reproducción generacional: Se reemplaza la población completa en cada iteración.

- Reproducción *steady-state*: Reemplaza sólo unos pocos individuos en cada iteración.

- **Selección y operadores genéticos de cruce y mutación**

Selección

El propósito de la selección de padres es incrementar la probabilidad de reproducir miembros de la población que tengan buenos valores de la función objetivo.

Se citan a continuación algunos mecanismos de selección de individuos:

- Selección neutral. Se trata de una selección aleatoria, todos los individuos tienen la misma probabilidad de ser seleccionados.

- Mecanismos proporcionales al *fitness*. Cada individuo tiene una probabilidad p_i de ser seleccionado. Esta probabilidad viene dada por:

  $p_i = fitness\ (i) / Suma\ de\ fitness$

- Mecanismos basados en ranking. Cuando la función objetivo es negativa, o puede serlo, el método anterior no puede usarse. Este método ordena los individuos de la población en función de la calidad de la solución, para luego escoger una dentro de ese orden.

- Torneos. Se escogen dos o más individuos para luego escoger los mejores entre éstos.

Cruce

El cruce es el procedimiento por el cual dos individuos intercambian parte de su material genético para crear un nuevo individuo. En general, consiste en intercambiar trozos de cromosomas de los progenitores, por lo que los descendientes se parecen a los progenitores. Para problemas con una codificación directa, pueden obtenerse descendientes no admisibles, siendo necesario cruces específicos para el problema.

Mutación

Es la introducción de un cambio aleatorio en un gen de un individuo. La mutación introduce diversidad de los descendientes respecto a los progenitores. En el caso de codificación binaria se puede cambiar el valor de un bit con una cierta probabilidad.

Consiste en pasar de un “0” a un “1” o viceversa. En el caso de codificación con variables de permutación, la mutación podría consistir en el intercambio de información que hay en dos posiciones distintas del individuo.

Cuando se dispone de más de un operador, hay que establecer la forma de elegir el operador que se usa en cada iteración para la reproducción. La forma más normal es una elección por ruleta, en función de una probabilidades de ejecución. Suele asignarse mayor probabilidad al cruce que a la mutación con objeto de guiar un procedimiento más intensivo en la búsqueda.

- **Criterios de eliminación de individuos de la población**

Este es otro criterio a determinar cuando se diseña un Algoritmo Genético. Algunos de los criterios más usados son:

- Mecanismos proporcionales al *fitness*. Cada individuo tiene una probabilidad p_i de ser seleccionado. Esta probabilidad viene dada por:

  $$1\text{-}p_i = 1\text{-}(fitness_i / Suma\ de\ fitness)$$

  Donde $fitness_i$ es el *fitness* del individuo que se está estudiando y *Suma fitness* es la suma de los *fitness* de todos los individuos de la población.

- Mecanismos basados en ranking: Se ordena los individuos de la población en función de la calidad de la solución, para luego escoger con una probabilidad entre los peores.

- Escoger el peor individuo entre todos los que formen la población.

- **Criterios de parada del algoritmo**

Los criterios más comunes de finalización del algoritmo son:

- Un número determinado de iteraciones.

- Cuando la mejor solución no mejora tras un cierto número de iteraciones.

Capítulo 4

Escenario I: El problema PDP con instantes fijos de entrega y una planta de producción

Índice

4.1. INTRODUCCIÓN

En este escenario consideramos el problema asociado a la planificación conjunta de la producción y distribución de pedidos (PDP) cuando los pedidos tienen un instante fijo de entrega y deben ser preparados en una única planta con capacidad limitada. El problema se define como la selección de pedidos para ser procesados en una planta de fabricación e inmediatamente distribuidos a la localización del cliente.

Recordemos los supuestos que se admiten en este escenario:

- El instante de entrega del pedido es fijo, no pudiendo servirse fuera de ese instante.

- Consideramos una planta de fabricación con capacidad limitada *C*, desde la que los vehículos parten y donde se encuentran inicialmente. La producción de un pedido se considera un proceso continuo que requiere una unidad de capacidad durante su tiempo de proceso. Si consideramos que la planta tiene *C* unidades de capacidad, entonces a lo sumo *C* pedidos podrían ser preparados simultáneamente.

- Para el reparto se cuenta con un conjunto de vehículos con las mismas características. Cuando se prepara un pedido en la planta, éste es inmediatamente distribuido (no espera) al lugar donde se localiza el cliente que cursó la petición.

- Asumimos que el tamaño de pedido es menor que la capacidad del vehículo.

- Un vehículo no puede distribuir simultáneamente más de un pedido. Cuando un pedido se descarga, el vehículo retorna a la planta de producción y queda disponible para otro envío.

- El número y valor de los pedidos, sus tiempos de proceso y tiempos de viaje son conocidos y fijos.

- También se asume que un pedido no puede ser interrumpido en ningún instante de sus dos fases de procesamiento.

El estudio del problema PDP en este escenario se estructura en dos apartados. En el apartado 4.2 se formula el problema mediante programación lineal entera. En el apartado 4.3 se identifican dos variantes del problema, que se distinguen por el número de vehículos disponibles. Estas variantes definen dos nuevos escenarios del problema que se estudian en los capítulos siguientes.

4.2. FORMULACIÓN DEL PROBLEMA

A partir de la notación establecida en el capítulo 2, los datos relativos a cada pedido i del conjunto de pedidos solicitados $P = \{1,...,n\}$ se resume en la figura 4.1:

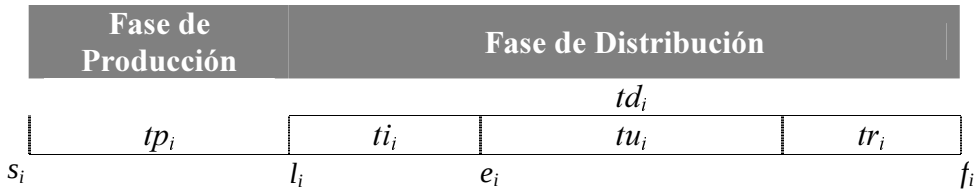


Figura 4.1. Datos asociados al pedido i en el Escenario I

El problema considera una capacidad finita de C unidades y un conjunto de V vehículos. Asumimos, sin pérdida de generalidad que:

a. Todos los datos numéricos son enteros positivos

b. $s_1 \leq s_2 \leq \ldots \leq s_n$

Sea $S(t) = \{i \in P: s_i \leq t \ \& \ l_i > t;\ t \in [0...T]\}$ el conjunto de pedidos cuyas fases de producción se solapan en el instante t.

Para formular el problema consideraremos la formulación del problema FSP descrita en [Fischetti, 1990]. Las variables del modelo se definen como sigue:

$$x_{ij} = \begin{cases} 1 & \text{si el vehículo } j \text{ sirve el pedido } i \\ 0 & \text{en otro caso} \end{cases}$$

$$p_i = \begin{cases} 1 & \text{si el pedido } i \text{ es servido} \\ 0 & \text{en otro caso} \end{cases}$$

$$i = 1 \ldots n \quad j = 1 \ldots V$$

Restricciones:

- Integridad entre las variables x_{ij} y p_i

$$\sum_{j=1}^{V} x_{ij} = p_i \qquad i = 1 \ldots n$$

- Capacidad limitada en la planta. Para asegurar que al mismo tiempo no se estén procesando más pedidos de los que indica la capacidad de fabricación de la planta se impone:

$$\sum_{i \in S(t)} p_i \leq C \quad t = 0 \ldots T$$

- Para imponer que dos pedidos, cuyas actividades coinciden en algún instante de tiempo, no puedan ser distribuidos por el mismo vehículo, se debe hacer previamente la distinción ya apuntada de la posible participación del vehículo en el proceso productivo.

  Atendiendo a esta circunstancia, si el vehículo encargado de la distribución del pedido participa también en el proceso de fabricación, debe estar disponible en el instante de comienzo de la fase de producción s_i. La restricción se expresa de la forma:

  $$x_{ij} + x_{kj} \leq 1 \qquad i = 1 \ldots n-1;\ j = 1 \ldots V;\ \{k > i : s_k < f_i\}$$

  Si por el contrario, el vehículo no interviene en la fabricación del pedido que distribuye, no tiene que estar disponible hasta el instante l_i en el que comienza la fase de distribución. La restricción quedaría entonces como:

  $$x_{ij} + x_{kj} \leq 1 \qquad i = 1 \ldots n-1;\ j = 1 \ldots V;\ \{k > i : l_k < f_i\}$$

La función objetivo maximiza el valor total de los pedidos servidos.

El modelo general que considera la no-participación del vehículo en el proceso productivo quedaría como sigue:

$$Max \quad \sum_{i=1}^{n} w_i p_i$$

sujeto a

$$\sum_{j=1}^{V} x_{ij} = p_i \qquad i = 1 \ldots n \qquad (1)$$

$$x_{ij} + x_{kj} \leq 1 \qquad i = 1 \ldots n-1;\ j = 1 \ldots V;\ \{k > i : l_k < f_i\} \qquad (2)$$

$$\sum_{i \in S(t)} p_i \leq C \qquad t = 0 \ldots T \qquad (3)$$

$$x_{ij} \in \{0,1\} \quad p_i \in \{0,1\}$$

4.3. ANÁLISIS DE VARIANTES DEL PROBLEMA

El problema PDP puede ser resuelto a través de diferentes estrategias que dependen del valor de algunos de sus parámetros, en concreto, de la capacidad de producción C y el número de vehículos V. Una condición necesaria y suficiente para la existencia de una solución en la que se sirvan todos los pedidos viene dada por el siguiente lema:

Lema: *Una solución incluyendo todos los pedidos existe si y sólo si: a) el máximo grado de simultaneidad en la planta es menor o igual a la capacidad C de la planta; y b) el máximo grado de simultaneidad en la fase de distribución es menor o igual al número V de vehículos disponibles.*

Supongamos n pedidos para procesar en el intervalo $[0,T]$, donde cada pedido i se representa por la tripleta (s_i, l_i, f_i). El máximo grado de simultaneidad en la fase de producción L^P y en la fase de distribución L^D se definen de la siguiente forma:

L^P = máximo$\{L^P_t: 0 \le t \le T\}$ con $L^P_t = \{i \in P: s_i \le t \le l_i\text{-}1\}$

L^D = máximo$\{L^D_t: 0 \le t \le T\}$ con $L^P_t = \{i \in P: l_i \le t \le f_i\text{-}1\}$

Este lema es una consecuencia del lema de [Kroon, 1995] sobre el problema FSP.

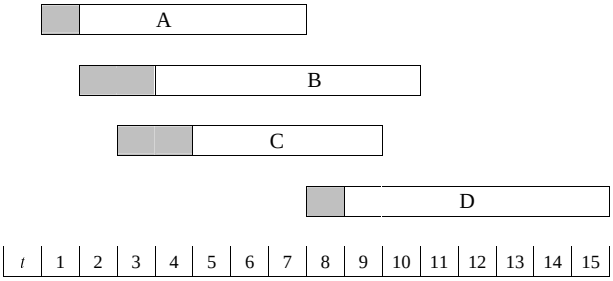


Figura 4.2. Plan cronológico para un ejemplo con 4 pedidos

La figura 4.2 representa un ejemplo con 4 pedidos. La porción sombreada representa la fase de producción y la porción transparente la fase de distribución. En este ejemplo L^P es igual a 2 y L^D es igual a 3. Es claro que sólo si $C \ge 2$ y $V \ge 3$ podrían todos los pedidos ser servidos. Por tanto, la capacidad de producción mínima para procesar todos los pedidos sería 2 y el número mínimo de vehículos para servirlos 3.

Si sólo uno de los dos grados de simultaneidad excede a alguno de los parámetros C o V, el problema llega a ser equivalente al problema de programación de trabajos fijos (FSP). Nótese, por ejemplo, que si L^D fuera menor que el número de vehículos existentes, entonces podríamos despreciar la fase de distribución de los pedidos porque siempre habría vehículos disponibles. En este caso, asimilando pedidos a trabajos y capacidad C a número de máquinas, el problema se transforma en encontrar un subconjunto de trabajos con valor total máximo que pueda ser procesado por las máquinas disponibles, es decir, el problema Max. FSP. Si ambos, L^P y L^D exceden la capacidad y el número de vehículos, respectivamente, entonces el problema llega a ser más interesante y complejo.

Según ello, el estudio y resolución del problema PDP en este escenario va a considerar los siguientes dos casos:

- Problema PDP con limitación en los recursos de producción y distribución. Se considera una capacidad de producción y un número de vehículos fijos, que podrían ser menores a L^P y L^D. Denominamos este caso como PDP con número limitado de vehículos, presentado como Escenario IA del problema.

- Problema PDP con limitación única en los recursos de producción. Se considera, por tanto, un número ilimitado de vehículos. Denominamos este caso como PDP con número ilimitado de vehículos, presentado como Escenario IB del problema.

Capítulo 5

Escenario IA: Número limitado de vehículos

Índice

5.1. INTRODUCCIÓN

Cuando en el problema se impone una capacidad de producción limitada *C* y un número de vehículos fijo *V*, el problema PDP puede ser considerado como un caso especial de sistema de flujo uniforme con máquinas en paralelo (FSPM). El problema PDP en este escenario corresponde con un problema FSPM con dos estaciones de máquinas, donde los pedidos equivalen a trabajos. La primera estación constaría de *C* máquinas y la segunda de *V* máquinas. Además, se asume *no-wait* entre procesos, puesto que la fase de distribución comienza justo después del fin de la fase de producción.

Como ya se ha comentado, la mayoría de los problemas FSPM no consideran fechas de entrega, y en la inmensa mayoría el objetivo es minimizar el *makespan*, es decir, el tiempo requerido para procesar todos los trabajos. Los trabajos que consideran fechas de entrega asumen el procesamiento de todos los pedidos. El problema PDP en este escenario es sin duda novedoso porque representa un problema jamás estudiado. Podemos considerar nuestro problema como un problema FSPM cuyo objetivo es maximizar el peso de los trabajos entregados justo en su fecha de entrega, asumiendo instantes de llegada al sistema.

La complejidad del problema PDP en este escenario, obliga a la aplicación de algoritmos aproximados para su resolución. Sin embargo, previamente desarrollamos un método de solución exacta, con un doble objetivo:

- Analizar la complejidad del problema para la obtención de soluciones exactas.

- Proveer de soluciones exactas a una batería de problemas para un posterior estudio de la bondad de los métodos aproximados que serán implementados.

5.2. MÉTODO EXACTO: GRAFO DE ESTADOS ADMISIBLES

El método que a continuación se desarrolla parte de la idea expresada por [Arkin y Silverberg, 1987] para la resolución exacta del problema FSP con varias clases de máquinas y trabajos. Igual que en dicho método, en nuestro procedimiento proponemos un algoritmo de flujo a coste mínimo en un grafo dirigido *G*. El grafo *G* recoge las posibles soluciones admisibles en el problema. Cada solución corresponde con un camino desde el nodo inicial al nodo final del grafo. La construcción del grafo está basada en un sencillo método de evaluación de estados admisibles en la planificación de los pedidos. La dinámica del proceso de construcción del grafo se aleja de la programación dinámica porque la información

que se almacena en cada nodo no permite conocer cómo se ha llegado hasta él. Cada nodo representa un estado admisible en la planificación de los pedidos, es decir, un conjunto de pedidos procesados de forma admisible. Los estados que pueden conectar con un estado cualquiera pueden ser varios, por lo que una estrategia basada en programación dinámica tendría que individualizar cada posible situación en la planificación de los pedidos, teniendo en cuenta el proceso llevado desde el comienzo de la planificación. Ello conllevaría un aumento notable del número de nodos en el grafo. Experimentos preliminares no recogidos en esta investigación apuntan a la conveniencia de descartar el uso de programación dinámica sobre esta clase de problemas.

El proceso de construcción del grafo puede ser descrito como sigue: Partimos inicialmente del conjunto de pedidos ordenado por el instante de inicio, es decir, $s_1 \leq s_2 \leq \ldots \leq s_n$. El conjunto $N_i = \{ s_1^i, s_2^i, \ldots, s_m^i \}$ $i = 1 \ldots n$, se usa para representar todos los posibles estados admisibles que incluyen el procesamiento en último lugar del pedido i. Cada estado s_r^i está compuesto de un conjunto admisible de pedidos que pueden ser simultáneamente procesados junto al pedido i. Cada estado corresponderá a un nodo del grafo. También se incluye un estado inicial s_o sin actividad y un estado final de la planificación s_f. Denotamos $N_0=\{s_0\}$ y $N_{n+1}=\{s_f\}$. Para determinar el conjunto de estados N_i, $i = 1...n$, se evalúa la admisibilidad de procesar el pedido i junto con todos los pedidos pertenecientes a cada uno de los estados precedentes, es decir, estados pertenecientes a $N_0, N_1,...,N_{i-1}$.

Para un estado s_r^i creado a partir del estado s_u^k, $k<i$, el conjunto de pedidos de s_r^i se define como:

$$O(s_r^i) = \{i\} \cup \{ j \in O(s_u^k): f_j > s_i;\ i, j, k \in P\}$$

$O(s_r^i)$ contiene los pedidos pertenecientes al estado s_u^k que continúan procesándose en el instante de comienzo del pedido i, es decir, en el instante s_i. El pedido i se incluye también en el conjunto $O(s_r^i)$. El máximo grado de simultaneidad en la planta y en la fase de distribución para el conjunto de pedidos de s_r^i viene definido de la siguiente forma:

$$L^P[O(s_r^i)] = \text{máximo}\{ L_t^P [O(s_r^i)]: 0 \leq t \leq T\} \text{ con } L_t^P [O(s_r^i)] = \{j \in O(s_r^i): s_j \leq t \leq l_j - 1\}$$

$$L^D[O(s_r^i)] = \text{máximo}\{ L_t^D [O(s_r^i)]: 0 \leq t \leq T\} \text{ con } L_t^D [O(s_r^i)] = \{j \in O(s_r^i): l_j \leq t \leq f_j - 1\}$$

Estas medidas se usan para determinar la admisibilidad de los estados s_r^i. El procedimiento de evaluación y creación de un estado viene descrito en el siguiente algoritmo:

Procedimiento ConstruccionEstado (Pedido *i*)
  Para cada estado $s^k_u \in N_k$ $k = 0 \ldots i\text{-}1$
    $r = r + 1$
    Crea Estado s^i_r (i, s^k_u)
    Si $(L^P[O(s^i_r)] \leq C)$ & $(L^D[O(s^i_r)] \leq V)$ **entonces**
      {*estado admisible*}
      Si $s^i_r \in N_i$ **entonces**
        {el estado coincide con algún estado ya existente en *i*}
        Crea Arco (s^k_u, s^i_r, w_i)
      Si no
        {*nuevo estado para el pedido i*}
        $N_i = N_i$ & s^i_r
        Crea Arco (s^k_u, s^i_r, w_i)
      Fin Si
    Fin Si
  Fin Para
Fin

Crea Estado crea un nodo en el grafo que contiene los pedidos pertenecientes a s^i_r. Crea Arco(*nodo origen*, *nodo destino*, *peso*) crea un arco en *G* que representa la adición del pedido *i* al plan formado por los pedidos pertenecientes al estado representado en *nodo inicial*. El peso asignado es el valor del pedido *i*.

Por tanto, existe un nodo en el grafo por cada estado admisible del problema. De la misma forma, cualquier camino de s_0 a s_f representa un plan admisible, y cada arco del camino representa el procesamiento de un nuevo pedido.

Finalmente y para obtener el grafo completo, necesitamos unir cada nodo terminal del grafo al nodo final s_f. El peso de estos arcos es cero.

El camino más largo desde s_0 a s_f representa la solución óptima del problema. La construcción realizada proporciona un grafo sin ciclos y directamente enumerado lo que facilita la obtención del camino máximo. El algoritmo utilizado para el cálculo del camino máximo en un grafo enumerado viene descrito en [Larrañeta, 1999].

Sea A^* el conjunto de arcos pertenecientes al camino solución. Los pedidos seleccionados para ser servidos serán aquellos pedidos *i* asociados con arcos $(s^k_u, s^i_r) \in A^*$.

La figura 5.2 muestra el grafo correspondiente al conjunto de pedidos representados en la figura 5.1. La capacidad considerada fue 1 y el número de vehículos 2.

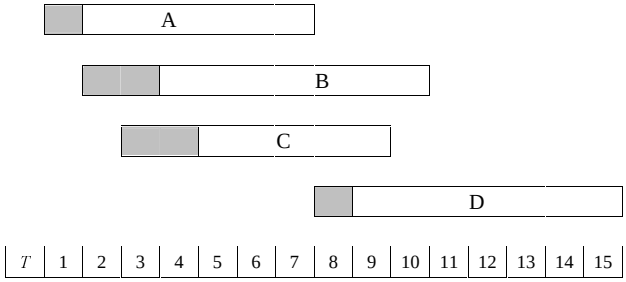


Figura 5.1. Plan cronológico para un ejemplo con 4 pedidos

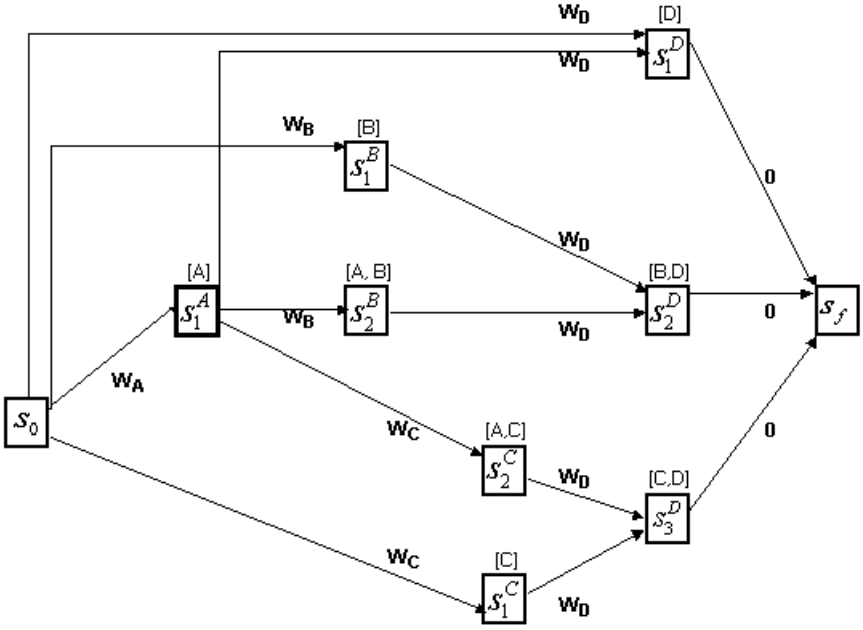


Figura 5.2. Grafo correspondiente al ejemplo de Figura 5.1

Sobre cada nodo aparecen escritos los pedidos que están siendo procesados en ese estado. Los conjuntos N_i , $i \in$ {A, B, C, D} vienen definidos como: $N_A = \{s^A_1\}$; $N_B = \{s^B_1, s^B_2\}$; $N_C = \{ s^C_1, s^C_2\}$; $N_D = \{ s^D_1, s^D_2, s^D_3\}$.

Asignando la siguiente tabla de pesos:

Pedidos	A	B	C	D
w_i	3	5	7	4

Tabla 5.1. Pesos para ejemplo de figura 5.1

El camino máximo aparece reflejado en la siguiente figura:

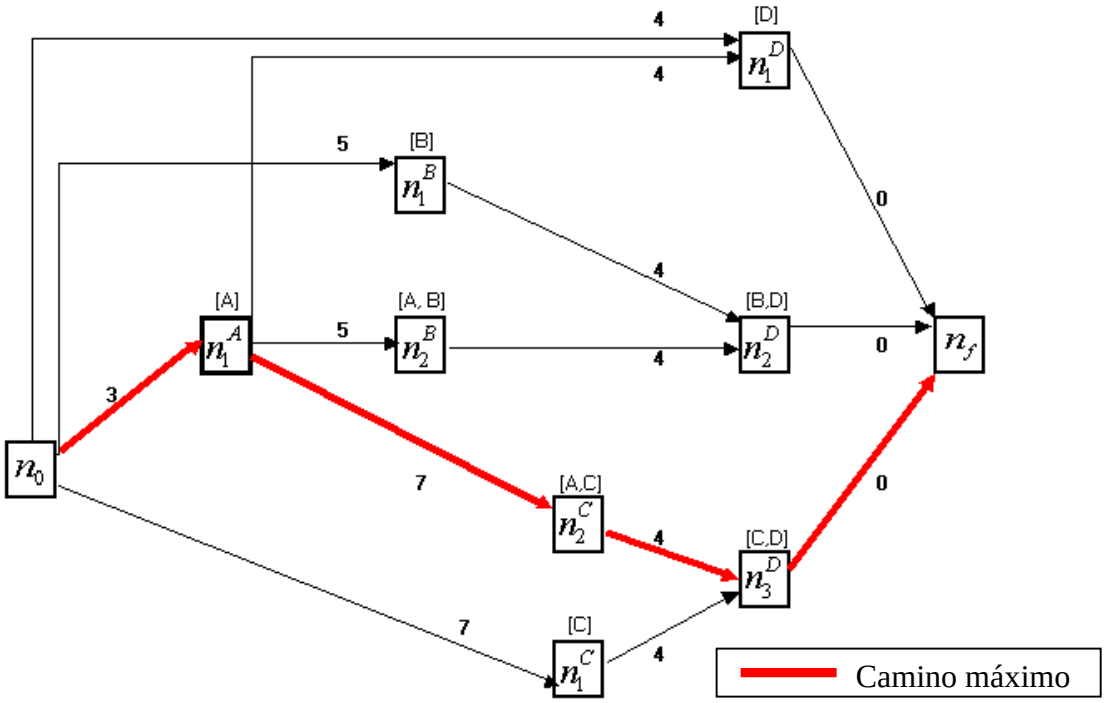


Figura 5.3: Camino máximo del grafo de figura 5.2

La solución óptima pasa por procesar los pedidos A, C y D.

5.2.1. Complejidad del proceso de construcción

En lo que sigue probaremos que la complejidad del proceso de construcción de G es de orden $O(((C + V) \log (C + V))\ n^{(C + V)})$. Para tal fin, calcularemos primero la complejidad computacional del proceso de construcción de cada nodo y finalmente, el número de nodos y arcos en el grafo.

1. Complejidad computacional del proceso de construcción de un estado

La complejidad en la formación de un estado se restringe al proceso de evaluación de admisibilidad, el cual consiste en determinar el máximo grado de simultaneidad de un conjunto de pedidos. [Hashimoto y Stevens, 1971] describen un algoritmo de orden $O(J\log(J))$ para determinar dicho grado a un conjunto de J trabajos. El conjunto de pedidos en cada estado es, a lo sumo, igual a la suma de C y V, debido a que en cualquier estado no pueden estar procesándose más pedidos que los que el número de vehículos y la capacidad de producción permiten. Por tanto, la complejidad es de orden $O((C + V) \log (C + V))$.

2. Número de nodos y arcos en G

Para el cálculo del orden del número de nodos y arcos, atenderemos a dos casos:

a) Siempre existen recursos disponibles para el procesamiento de todos los pedidos.

b) Caso genérico del Escenario IA: Los recursos C y V limitan el procesamiento de todos los trabajos.

Es obvio, que el problema en el caso a) no sería tratado a través de este método, por la sencilla razón de que la solución óptima es inmediata: se atienden todos los pedidos. Sin embargo, el cálculo es útil para determinar el orden en el número de nodos y arcos si no se ha realizado una comprobación previa de la limitación que imponen los recursos del problema.

a) Recursos disponibles para el procesamiento de todos los pedidos

Por cada conjunto de estados N_i, el máximo número de estados es $\sum_{k=0}^{i-1} |N_k|$, es decir, la suma de los estados de los conjuntos previos a i. La tabla 5.2 contiene estos valores:

N_i	N_0 (s_0)	N_1	N_2	N_3	N_4	…	N_n	N_{n+1} (s_f)
Número de estados en N_i ($\|N_i\|$)	1	1	2	4	8	…	2^{n-1}	1

Tabla 5.2: Máximo número de estados en cada conjunto N_i

Por tanto, el número total de nodos en G, n_G sería:

$$n_G = \sum_{i=1}^{n} \left(|N_i| \right) + |N_0| + |N_{n+1}| = \sum_{i=1}^{n} 2^{i-1} + 2 \equiv O(2^n)$$

Como la adición de un nodo provoca la adición de un arco, el número total de arcos sería también de orden $O(2^n)$.

b) C y V limitan el procesamiento de todos los pedidos

Para contar el número de nodos en G en el caso general del problema, nótese que los recursos en cada nodo podrían representarse como un vector de $C + V$ posiciones que ocupan los pedidos que hacen uso de los mismos. Veamos un ejemplo de las posibilidades que podrían darse con 2 pedidos, A y B, y valores $C = 1$, $V = 2$:

Planificación	$C = 1$	$V = 2$	
1	A	-	-
2	B	-	-
3	A	B	-
4	B	A	-
5	-	A	B
6	-	A	-
7	-	B	-
8	-	-	-

Tabla 5.3: Ejemplo de posibilidades de planificación

Fijándonos en la tabla 5.3, las distintas posibilidades de planificación en planta corresponden con la siguiente expresión:

$\sum_{c=1}^{\min\{C,n\}} \binom{n}{c} + 1$ (el término independiente 1 indica la situación en la que la planta se encuentra sin actividad).

El supuesto general contempla la limitación de recursos. Ello implica que el número de pedidos n sea mayor que C y V, puesto que en otro caso todos los pedidos serían servidos. Por ello, la expresión anterior puede simplificarse en la siguiente expresión:

$$\sum_{c=1}^{C} \binom{n}{c} + 1$$

Para el ejemplo serían tres posibilidades:

$$\sum_{c=1}^{1} \binom{2}{c} + 1 = \binom{2}{1} + 1 = 3$$ [procesar A, procesar B, y no procesar ningún pedido]

Por otro lado, sobre cada una de esas posibilidades, aparecen en la fase de distribución un número de nodos que corresponde con la siguiente expresión:

$$\sum_{v=1}^{\min\{V,p\}} \binom{p}{v} + 1$$

donde p expresa el número de pedidos n menos los pedidos que se estén procesando en ese momento en la planta, es decir, que ocupen alguna de la posiciones asociadas

al consumo de capacidad. Por tanto $p \in \{1,...,n\}$. El término independiente vuelve a indicar la situación sin actividad. Veamos los valores para el ejemplo de la tabla 5.3:

Caso: No procesar ningún pedido

$$\sum_{v=1}^{2}\binom{2}{v}+1=\binom{2}{1}+\binom{2}{2}+1=4 \text{ [Planificación 5, 6, 7 y 8]}$$

Caso: Procesar un pedido

$$\sum_{v=1}^{1}\binom{1}{v}+1=\binom{1}{1}+1=2$$

En el caso de procesar el pedido A, correspondería a las planificaciones 1 y 3. Para el pedido B, serían la planificación 2 y 4.

Sabiendo que el parámetro p puede variar desde 1 hasta n, y tomando el peor de los casos, el orden en el número de nodos respondería a la siguiente expresión:

$$\sum_{c=1}^{C}\left(\binom{n}{c}+1\right)\left(\sum_{v=1}^{\min\{V,p\}}\binom{n}{v}+1\right)=O\left(\sum_{c=1}^{C}n^{c}\sum_{v=1}^{V}n^{v}\right)=O\left(n^{C}n^{V}\right)=O\left(n^{C+V}\right)$$

El número de arcos viene a ser del mismo orden ya que cada estado proviene de un solo estado anterior, y por tanto, el número de nodos se correspondería con el número de arcos. En realidad, como se verá en la experimentación, el número de arcos aumenta sensiblemente respecto al número de nodos, debido a la duplicidad de nodos en cada evento.

La complejidad total del proceso de construcción de G sería la complejidad computacional del proceso de construcción de un estado multiplicado por el número de estados, es decir, $O(((C + V) \log(C + V))n^{C+V})$.

La solución exacta se obtiene con la aplicación de un algoritmo de ruta mínima sobre el grafo dirigido y acíclico G, el cual se resuelve en tiempo polinomial de orden $O(|N| \log |N| + |A|)$ [Freedman y Tarjan, 1984], siendo N el conjunto de nodos y A el conjunto de arcos.

5.2.2. Regla de reducción del grafo

Para acelerar el proceso de construcción de G y ahorrar espacio de almacenamiento, es posible descartar algunos arcos mediante la aplicación de la siguiente regla:

En la generación de estados y arcos para el pedido *i* descartar arcos (s^k_u, s^i_m) $\forall s^k_u \in N_k \;\; \forall s^i_m \in N_i \;\; k<i$ si $\exists j \in P$ con $f_k \leq s_j$ y $f_j \leq s_i$

Esta regla está basada en el hecho de que no es necesario considerar transiciones entre estados que no aprovechan el uso de los recursos. Analicemos la regla para el siguiente ejemplo:

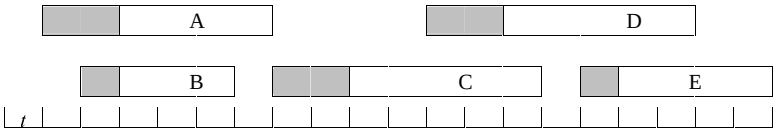


Figura 5.4: Ejemplo para regla de reducción

El grafo que resulta tras aplicar el procedimiento de construcción es el siguiente:

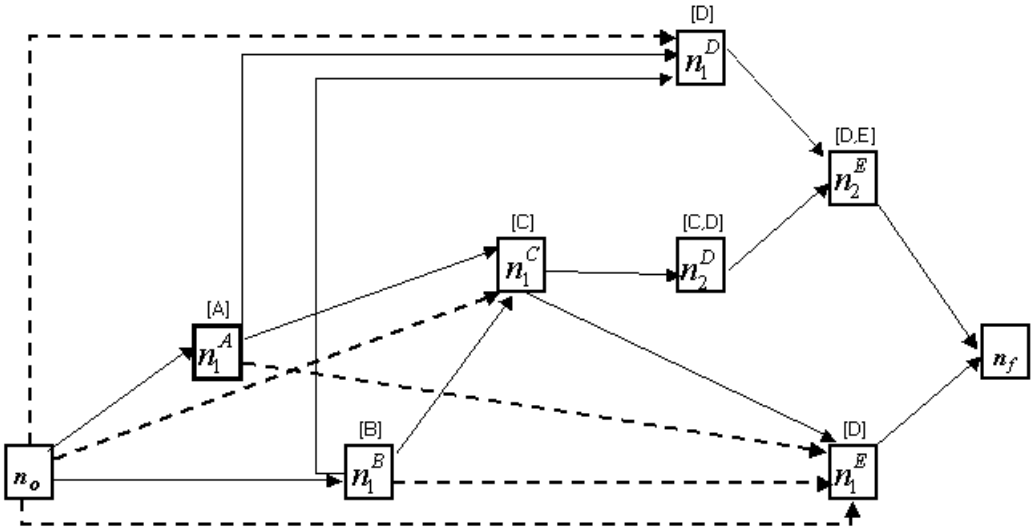


Figura 5.5: Grafo del ejemplo de la figura 5.4

Los arcos a trazos del grafo de la figura 5.5 serían arcos no necesarios y que se pueden eliminar. Si nos fijamos en la figura, el arco (n_1^A, n_1^E) representa la realización del pedido A y el pedido E. Esta transición entre estados está dominada, pues existe un camino alternativo $(n_1^A \rightarrow n_1^C \rightarrow n_1^E)$ que conlleva la realización además del pedido *C*. Igual ocurre con los arcos (n_1^B, n_1^E), (n_o, n_1^C), (n_o, n_1^D) y (n_o, n_1^E).

5.3. PROCEDIMIENTO GRASP PARA LA RESOLUCIÓN DEL PROBLEMA

Recordemos que un procedimiento GRASP está formado por una fase de construcción de una solución admisible y una fase de búsqueda local. Una solución admisible en el problema PDP está formada por un subconjunto de pedidos que cumplen las restricciones asociadas a la capacidad de producción C y al número de vehículos disponibles V.

1) Fase de construcción de una solución

En esta fase del GRASP se construye una solución admisible, un elemento en cada paso, guiado por una función voraz o avariciosa (*greedy*) y una elección de carácter aleatorio.

La función *greedy* para cada pedido, definida como y_j , es igual al peso de los pedidos, $y_j = w_j$. Usamos por tanto una función sin carácter adaptativo, luego no se requiere el cálculo en cada iteración de dicha función, ya que el peso de cada pedido no varía de una iteración a otra. Sea S el conjunto, inicialmente vacío, de los pedidos que ya han sido introducidos en la solución. En cada iteración de esta fase, I denota al conjunto de pedidos que no pueden pertenecer al conjunto S, denominados pedidos incompatibles.

Sean $L^P[S \cup \{i\}]$ y $L^D[S \cup \{i\}]$ el mayor grado de simultaneidad en las fases de producción y distribución de los pedidos pertenecientes a la solución S más el pedido i. El conjunto I está formado por los pedidos que satisfacen una de las siguientes condiciones:

a) $L^P[S \cup \{i\}] > C$

b) $L^D[S \cup \{i\}] > V$

Una vez que se haya elegido un pedido para formar parte de la solución hay que actualizar el conjunto de pedidos incompatibles I, para que se refleje la elección del nuevo pedido.

El proceso de construcción continuará mientras $I \cup S \neq$ P, siendo P el conjunto de todos los pedidos solicitados. En el momento en que no se puedan incorporar más pedidos a la solución, todos los pedidos de P estarán incluidos o bien en S o bien en I.

Queda por describir el mecanismo que se va a emplear para la construcción de la lista restringida de candidatos (LRC). En principio todos los pedidos que no

pertenezcan ni al conjunto *I* ni al conjunto *S* pueden ser aceptados en la lista LRC, pero en lugar de hacer una elección avariciosa, se permite estar en la lista LRC si el peso del pedido está dentro de un porcentaje (α) del pedido con mayor y_i. Por lo tanto, α es el parámetro que restringe los candidatos ($0 \leq \alpha \leq 1$). Un pedido *j* es un candidato si $y_j \geq \alpha y^*$, con y^*=Máximo $\{y_i / i \notin S \ \& \ i \notin I\}$. Hay que hacer notar que si el parámetro α toma el valor 0, todos los pedidos compatibles forman parte de LRC, por lo que el procedimiento es totalmente aleatorio. Sin embargo, si α = 1, el GRASP se convierte en una heurística puramente voraz.

A continuación se muestra un pseudocódigo de la construcción de la lista LRC.

```
Procedimiento Construcción LRC (S, I, P, LRC, α)
    LRC = 0
    y* = max {yj : j∈P \ I∪S}
    Para j∈P \ I∪S
        Si yj ≥ αy* entonces
            LRC = LRC ∪ {j}
        Fin Si
    Fin Para
Fin
```

El pseudocódigo de la fase de construcción del GRASP para el problema PDP es el siguiente:

```
Procedimiento Construcción (P)
    S = ∅
    I = ∅
    Mientras I∪S ≠ O
        Construcción LRC (S, I, P, LRC, α)
        i = Elegir Aleatoriamente Elemento(LRC)
        S = S + {i}
        Actualización Conjunto Pedidos Incompatibles(S, I, P)
    Fin Mientras
Fin
```

2) Fase de búsqueda local.

Para un problema determinado, un algoritmo de búsqueda local actúa de un modo iterativo reemplazando la solución actual por la mejor de la vecindad respecto a una función de coste. Este algoritmo termina cuando no encuentra ninguna solución mejor en la vecindad.

La búsqueda local implementada en este GRASP se basa en un procedimiento de intercambio, donde un pedido que pertenece al conjunto de pedidos de la solución, es reemplazado por otro u otros que no están en la solución. Un pseudocódigo del procedimiento de búsqueda local se puede expresar como sigue:

Procedimiento Búsqueda Local (S^0, C, S^*)
Repetir
  $S^* = S^0$
  Para cada $j \in S^0$
    $S^B = S^0$
    B_j = Mejor_Intercambio (j, S^0, I)
    Si $B_j \neq \varnothing$ **entonces**
      $S_j = (S^B - \{j\}) \cup B_j$
      Si $w(S_j) > w(S^*)$ **entonces**
        $S^* = S_j$
      Fin Si
    Fin Si
  Fin Para
  Si $S^* <> S^0$ **entonces**
    Continuar=TRUE
    $S^0 = S^*$
  Si no
    Continuar= FALSE
  Fin Si
Hasta que Continuar=FALSE
Fin

El procedimiento *Mejor_Intercambio(j,*S^0*,I)* busca el conjunto de pedidos pertenecientes a *I* que pueden formar parte de la solución, una vez sacado de ella el pedido *j.* Para entender mejor este algoritmo de búsqueda local se analiza el siguiente ejemplo:

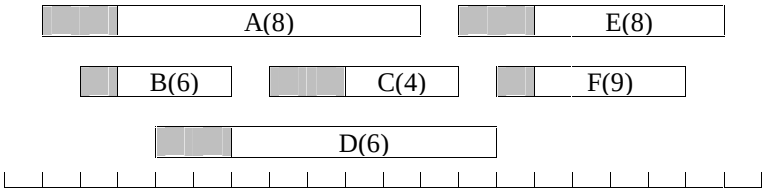


Figura 5.6: Ejemplo de la fase de búsqueda local (entre paréntesis, peso w_i)

En el ejemplo se considera una capacidad C = 1 y un número de vehículos V = 2. Si la solución inicial, obtenida en la fase de construcción, es S^0 ={A,D,E}, el peso total es $w(S^0) = w_A + w_D + w_E = 22$. Para calcular *Mejor_Intercambio*(A, {A, D, E},{B, C, F}), se evalúan los pedidos pertenecientes a I cuyos procesos de producción coincidan en algún instante con el periodo de tiempo en el que se fabrica el pedido A. En el ejemplo propuesto sería el pedido B. El proceso de evaluación consiste en añadir este pedido a la solución S^0 y examinar si es posible añadir más pedidos del conjunto I. En el ejemplo, es posible añadir el pedido C, junto con el pedido B. Estos pedidos formarían el conjunto B_j. El conjunto S_j es el resultado de incluir los pedidos de B_j en S^0 y quitar el pedido j, esto es, $S_j = (S^o \cup B_j) - \{j\}$. Cuando B_j es distinto del conjunto vacío y el conjunto S_j tiene un valor total mayor que el peso de la solución en vigor S^*, el conjunto S_j se convierte en la nueva solución en vigor S^*.

Veamos el procedimiento de búsqueda local completo para el ejemplo de la figura 5.6:

S^O	$W(S^O)$	I					
{A,D,E}	22	{B,C,F}					
			j	B_j	S^B	$w(S^B)$	S^*
			A	B,C	{B,C,D,E}	24	$S^*=S^B$
			D	C	{A,C,E}	20	-
			E	F	{A,D,F}	23	-
$S^O=S^*$	$W(S^O)$	I					
{B,C,D,E}	24	{A,F}					
			j	B_j	S^B	$w(S^B)$	S^*
			B	∅	-	-	-
			C	∅	-	-	-
			D	∅	-	-	-
			E	F	{B,C,D,F}	25	$S^*=S^B$
$S^O=S^*$	$w(S^O)$	I					
{B,C,D,F}	25	{A,E}					
			j	B_j	S^B	$w(S^B)$	S^*
			B	∅	-	-	-
			C	∅	-	-	-
			D	∅	-	-	-
			F	E	{B,C,D,E}	24	-
S^*={B,C,D,F}	$w(S^O)$= 25						

5.4. ALGORITMO DE BÚSQUEDA TABÚ

Con objeto de mejorar el procedimiento de búsqueda definido en el método GRASP, se ha desarrollado un algoritmo de búsqueda tabú que permite una búsqueda más exhaustiva del espacio de soluciones. El algoritmo GRASP anteriormente implementado adolece de una propiedad en el procedimiento de búsqueda local. Esta propiedad no es otra que la de no permitir la disminución en el número de pedidos pertenecientes a una solución. Si nos fijamos en el ejemplo descrito anteriormente, podemos comprobar que mediante el tipo de sustituciones empleado, el número de elementos de una solución a otra puede ser el mismo o mayor, pero nunca menor. Este hecho descarta soluciones de tamaño *t*, siempre que me encuentre en una solución con tamaño mayor que *t*. La búsqueda tabú que se ha desarrollado corrige el procedimiento de búsqueda para permitir la disminución en el número de elementos de una solución.

Como ya se ha comentado, la búsqueda tabú es un procedimiento heurístico que explora el espacio de soluciones a través de sucesivos movimientos desde una solución a la mejor de sus vecinas. Los elementos que definen un algoritmo de búsqueda tabú son:

- Una solución inicial
- Movimientos y estructura de la vecindad
- Lista tabú
- Criterio de parada
- Criterio de aspiración

❑ Solución inicial

Una solución inicial del algoritmo puede ser cualquier combinación de pedidos que no viole ninguna de las restricciones del problema, esto es, que los pedidos se entreguen en su fecha fijada sin existir demora entre los procesos de producción y distribución de cada pedido servido, además de que el número de pedidos en proceso de fabricación y distribución en cada instante no supere la capacidad de producción de la planta y el número de vehículos disponibles, respectivamente.

Los procedimientos usados para obtener una solución inicial han sido tres:

- Elección aleatoria de los pedidos que se van incorporando a la solución.

- Estrategia avariciosa de elección según el peso de los pedidos. Tanto en esta estrategia como en la anterior el proceso de creación de la solución finaliza cuando no es admisible la introducción de ningún pedido más.

- Soluciones proporcionadas por el algoritmo GRASP del apartado anterior, únicamente realizando una iteración del método.

Estas tres estrategias servirán para medir la sensibilidad del algoritmo respecto a este parámetro.

- **Movimientos y estructura de la vecindad**

Denotemos por *S* el conjunto que contiene los pedidos que pertenecen a la solución actual. Una solución vecina de *S* se obtiene por un procedimiento de cambio donde un pedido *i* perteneciente al conjunto solución es reemplazado por otro/s pedido/s no pertenecientes al mismo. Dado un pedido $i \in S$, consideramos todos los pedidos *j*, $j \notin S$, cuya fase de producción o de distribución coincide en algún instante con la del pedido *i*. Usemos H_i para denotar el conjunto de esos pedidos. Los pedidos pertenecientes a H_i son primero ordenados por valor y tentativamente introducidos en el conjunto solución *S* para chequear si las restricciones del problema continúan cumpliéndose. La figura 5.7 ilustra un ejemplo para una instancia con $C = 1$ y $V = 2$. Entre paréntesis aparece el valor del pedido y en tono más oscuro los pedidos pertenecientes a cada solución. En la solución actual del ejemplo se sirven los pedidos A, D y E, mientras que en la solución vecina se sirven los pedidos B, C, D y E.

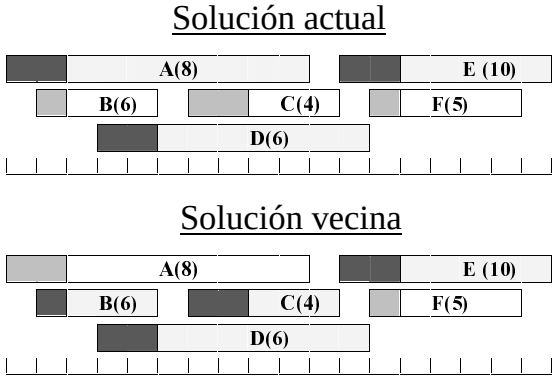


Figura 5.7: Movimiento de sustitución del trabajo A.

Los pedidos que entran en la solución pasan a ser tabú durante un número determinado de iteraciones.

Para permitir la extracción de pedidos de la solución sin introducir ninguno, cada cierto número de iteraciones se considera el movimiento de extraer un pedido de la solución. Estos pedidos pasan a formar parte de una segunda lista tabú que se actualiza cada vez que se realiza este tipo de movimiento. Este tipo de movimiento

lleva, lógicamente, a soluciones peores. Sin embargo, llega a ser útil para escapar de máximos locales y diversificar la búsqueda.

- **Lista tabú**

El tamaño de la lista tabú es un parámetro importante en un algoritmo de búsqueda tabú. En un primer diseño del algoritmo se optó por un tamaño de lista variable que permitiese medir la frecuencia con la que se aceptaban movimientos de extraer sin introducir, es decir, movimientos que disminuyen el número de elementos de la solución. Para ello se consideraba tabú un pedido que salía de la solución. Cuando se llegaba a una solución en la que todos los pedidos que no pertenecían a la misma eran tabú, la única opción era sacar un pedido sin introducir ninguno. El estudio realizado con esta implementación concluyó que era conveniente que este movimiento no se realizase con una frecuencia menor al doble del número de pedidos del problema, por lo que se estableció este valor como la frecuencia exacta para realizar un movimiento de extracción. En la configuración final del algoritmo, se cambió el concepto de movimiento tabú, pasando a ser tabú un pedido que se introducía en la solución. Para medir la duración de los elementos en la lista tabú, se realizó un análisis de sensibilidad que aparece reflejado en la experimentación de este capítulo.

- **Criterio de aspiración**

Para el criterio de aspiración usamos la forma más clásica en la que un movimiento tabú es aceptado si produce una solución mejor que la mejor solución obtenida hasta ese momento.

- **Criterio de parada**

El algoritmo se detiene después de un número máximo de iteraciones. Para el estudio del comportamiento del algoritmo se establecen 500, 1000 y 5000 iteraciones.

5.5. RESULTADOS COMPUTACIONALES

En este apartado se analizan los resultados proporcionados por la totalidad de los métodos desarrollados para el problema PDP en el escenario IA.

El primer apartado de la experimentación describe la obtención de las baterías de problemas. El apartado 5.5.2 presenta los resultados obtenidos con la aplicación del método de solución exacta. Se examina su complejidad con diferentes baterías de

problemas. También se evalúa la mejora ofrecida por la regla de reducción del grafo. Por último, en el apartado 5.5.3 se presentan los resultados de la aplicación de los algoritmos GRASP y Búsqueda Tabú. Todos los tiempos de computación vienen expresados en segundos sobre un Pentium III 850 Mhz.

Los porcentajes de error que aparecen en todos los experimentos relativos a los métodos heurísticos han sido calculados mediante la siguiente expresión:

$$\frac{(\text{Solución Óptima} - \text{Solución Aproximada})}{\text{Solución Óptima}} \times 100$$

5.5.1. Generación de problemas

La primera fase de los experimentos envuelve la construcción de 4 baterías de problemas. Los parámetros fundamentales en la construcción de los problemas fueron el número de pedidos y los promedios de simultaneidad de los pedidos en las fases de producción y distribución. El grado medio de simultaneidad mide la media del número de pedidos con actividad simultánea en las fases de producción y distribución. El cálculo de los promedios de simultaneidad de los problemas aparece descrito en el Anexo I de la tesis. Los tamaños considerados para los problemas fueron 25, 50, 75, 100 y 200 pedidos. Las siguientes tablas contienen los grados medios y mayores de simultaneidad en cada fase, los cuales se representan también en la figura 5.8:

$L^P{}_m$: Grado medio de simultaneidad en la fase de producción
$L^D{}_m$: Grado medio de simultaneidad en la fase de distribución
L^P: Mayor grado de simultaneidad en la fase de producción
L^D: Mayor grado de simultaneidad en la fase de distribución

	Número de pedidos							
	25				**50**			
	$L^P{}_m$	$L^D{}_m$	L^P	L^D	$L^P{}_m$	$L^D{}_m$	L^P	L^D
Batería 1	1.25	4.50	4.2	9.2	1.72	6.51	5.3	12.9
Batería 2	1.83	6.13	5.8	12.8	2.29	8.37	5.8	15.0
Batería 3	2.80	11.04	7.1	20.8	2.66	12.84	6.6	22.8
Batería 4	4.21	14.86	9.0	24.4	3.35	15.49	8.2	26.8

Tabla 5.5: Problemas de 25 y 50 pedidos

	Número de pedidos											
	75				**100**				**200**			
	$L^P{}_m$	$L^D{}_m$	L^P	L^D	$L^P{}_m$	$L^D{}_m$	L^P	L^D	$L^P{}_m$	$L^D{}_m$	L^P	L^D
Batería 1	2.04	7.96	5.8	14.4	2.14	8.81	5.9	15.8	1.88	8.51	6.2	13.4
Batería 2	2.47	9.33	6.7	16.6	2.56	10.06	6.5	17	2.2	11.6	6.9	18.5
Batería 3	2.62	13.47	5.8	22.2	2.6	13.80	5.7	21.5	2.46	13.75	7.2	22.5
Batería 4	3.05	15.12	7	24.0	2.87	15.13	6.5	24.7	2.74	15.11	7.0	23.3

Tabla 5.6: Problemas de 75, 100 y 200 pedidos

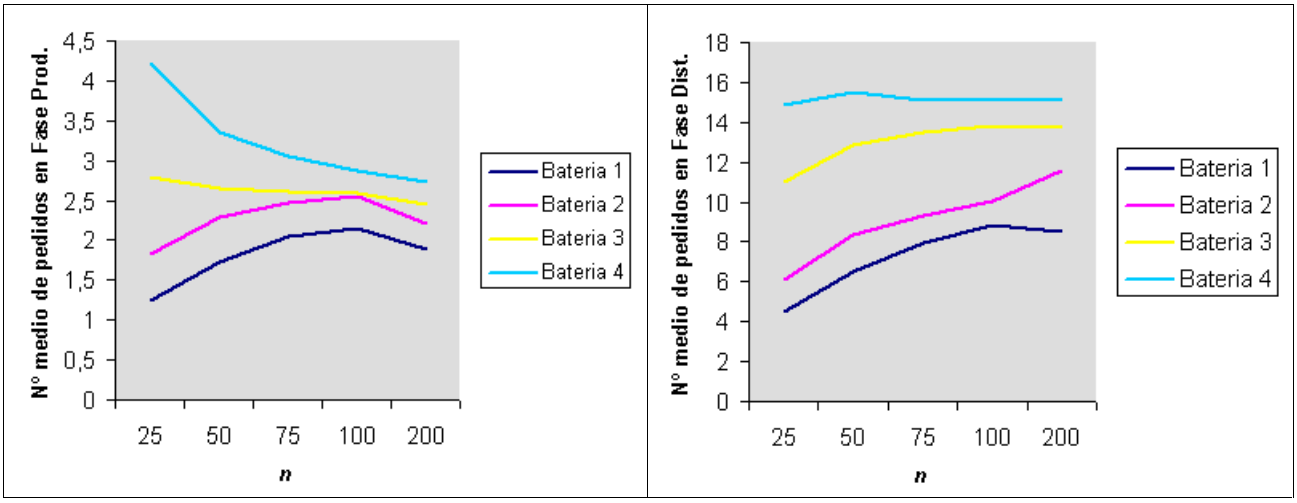


Figura 5.8: Grados medios de simultaneidad

Para cada tamaño se generaron 10 instancias diferentes, por lo que el número total de problemas considerados en cada batería fue de 50.

El tiempo de la fase de producción, fase de distribución y los pesos asignados a cada pedido fueron generados aleatoriamente dentro de los intervalos mostrados en la tabla 5.7.

	Intervalo
Producción (tp_i)	[1,5]
Distribución (td_i)	[4,24]
Peso (w_i)	[30,100]

Tabla 5.7: Intervalos de generación aleatoria

Según los datos de la tabla 5.7, la repercusión de la fase de distribución es bastante mayor que la de la fase de producción. La producción viene a ser entre un 15% y un 20% de la actividad completa del pedido.

En cuanto a la capacidad de fabricación *C* y el número de vehículos *V*, se tuvieron en cuenta distintas combinaciones de estos valores para la resolución de cada instancia. La tabla 5.8 refleja estos valores.

C	*V*
1	2
2	2
2	3
3	4

Tabla 5.8: Valores de *C* y*V*

5.5.2 Análisis del método exacto

El análisis del comportamiento del método de solución exacta se ha dividido en tres fases:

- En la primera fase se realiza el estudio de la evolución del tamaño del grafo respecto al tamaño del problema y al crecimiento del número de recursos *C* y *V*. (Sección 5.5.2.1)

- En la segunda fase se estudia el comportamiento de la regla de reducción del grafo. (Sección 5.5.2.2)

- La tercera fase presenta el comportamiento del método respecto al incremento en el grado de simultaneidad de los pedidos. (Sección 5.5.2.3)

Todos los resultados corresponden con los valores medios de las 10 instancias generadas para cada tamaño de problema. Los resultados se ofrecen agregados respecto al tamaño *n* del problema o al número de recursos empleados (*C*, *V*).

5.5.2.1. Evolución del tamaño del grafo

En este apartado se estudia el incremento del número de nodos, número de arcos y tiempo del algoritmo respecto al crecimiento del número de pedidos y del número de recursos. En la formación del grafo no se aplica la regla de reducción, pues como veremos en el tercer apartado del análisis del método, esta regla se ve influida por el grado de solapamiento de los pedidos. Los experimentos de este análisis se han realizado con la batería 3 de problemas. Los datos obtenidos se muestran en las tablas 5.9, 5.10 y 5.11. Estos mismos datos agregados respecto a *n* y (*C*,*V*) se representan gráficamente en las figuras 5.9, 5.10 y 5.11. Para mayor claridad en la tendencia exponencial, la mayoría de las gráficas se representan en escala logarítmica.

	25	50	75	100	200
(1,2)	242	657	998	1338	2574
(2,2)	302	760	1138	1519	2939
(2,3)	2067	6231	8896	11663	21181
(3,4)	10750	36202	47587	60427	103352

Tabla 5.9: Número medio de nodos

	25	50	75	100	200
(1,2)	696	8517	25656	52408	226523
(2,2)	871	9900	29343	59575	258414
(2,3)	6096	84827	232228	471367	1840789
(3,4)	31416	503572	1231012	2473714	8780113

Tabla 5.10: Número medio de arcos

	25	50	75	100	200
(1,2)	0.3	4.06	12.6	25.7	106.9
(2,2)	0.9	10.6	19.0	33.6	122.9
(2,3)	4.1	44.4	108.5	229.5	1331.8
(3,4)	41.1	697.2	1296.9	2883.3	29890.7

Tabla 5.11: Tiempo medio de ejecución

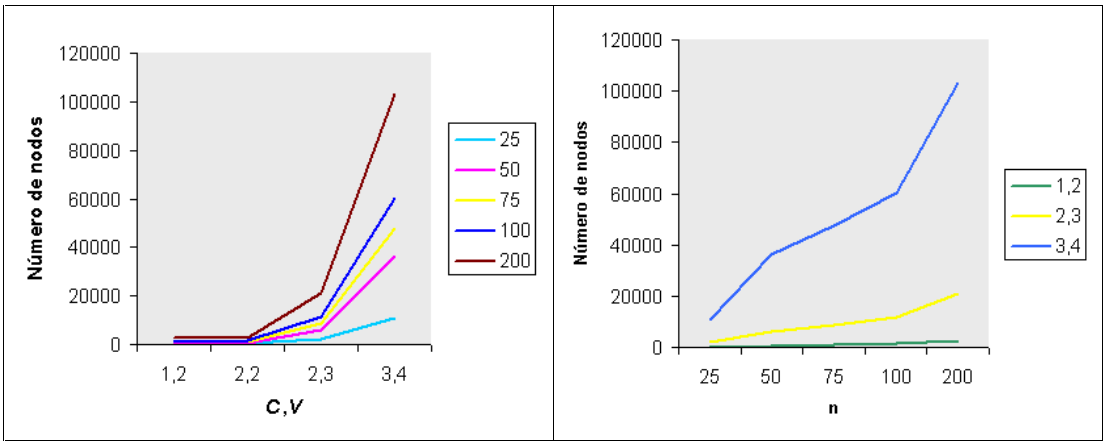


Figura 5.9: Número medio de nodos

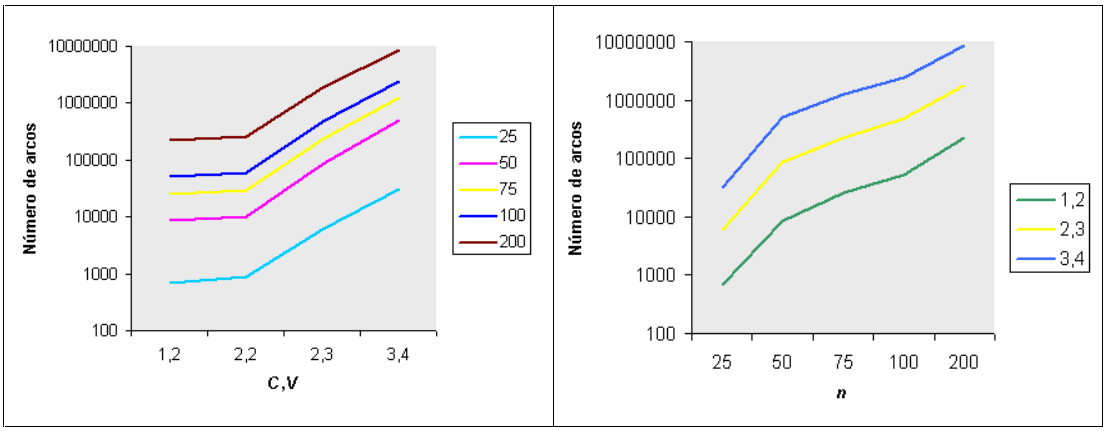


Figura 5.10: Número medio de arcos

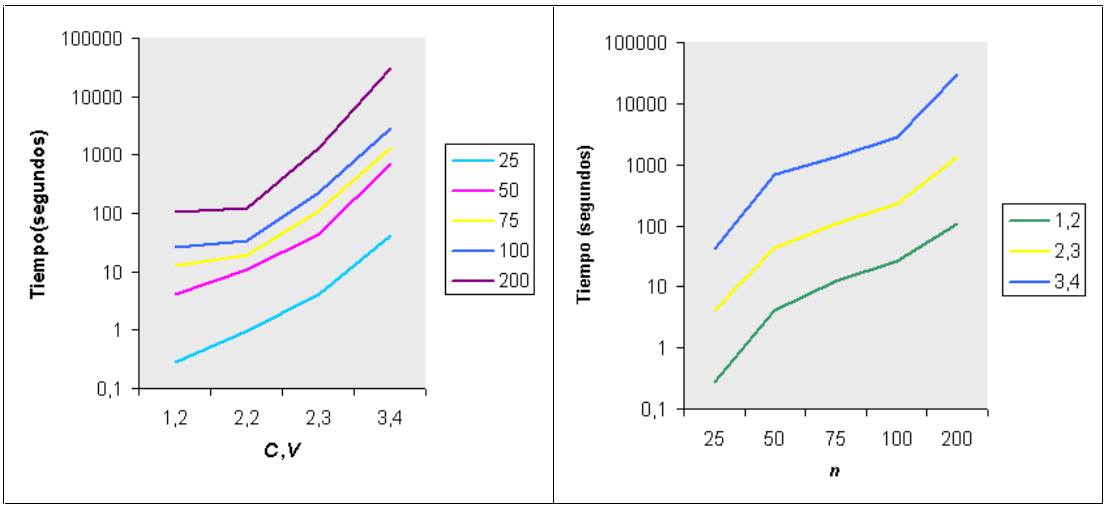


Figura 5.11: Tiempo medio de ejecución

A la vista de los datos mostrados en las tablas y figuras anteriores, se comprueba que los resultados responden al orden $O(n^{C+V})$ calculado para el número de nodos y arcos del grafo. El carácter exponencial se aprecia sobre todo con el aumento de los recursos, puesto que el número de estados admisibles aumenta sensiblemente.

5.5.2.2. Comportamiento de la regla de reducción

Para analizar los resultados obtenidos al aplicar la regla de reducción del grafo, se ha tomado como entrada la batería 3, obteniéndose los resultados presentados en las tablas 5.12 y 5.13. En este caso, no es necesario el estudio sobre el número de nodos puesto que la regla no afecta a este parámetro. Se contempla únicamente la reducción del número de arcos y los tiempos de ejecución.

	25	50	75	100	200
(1,2)	691	8128	18752	28596	59926
(2,2)	866	9383	21108	31980	66949
(2,3)	6091	83102	180942	276005	532380
(3,4)	31410	498243	995297	1508458	2692252

Tabla 5.12: Número de arcos con regla de reducción

	25	50	75	100	200
(1,2)	0.1	3.6	9.4	13.8	33.2
(2,2)	0.8	4.1	11.7	21.3	79.5
(2,3)	3.9	43.1	86.5	132.7	312.2
(3,4)	40.0	650.2	1063.6	1657.3	5122.6

Tabla 5.13: Tiempos de resolución con regla de reducción

A continuación se muestran gráficas comparativas del número de arcos y de los tiempos de ejecución obtenidos en la construcción del grafo con la regla de reducción y sin ella. También se representan los porcentajes de reducción del número de arcos. Los datos aparecen agregados respecto a n y (C,V). Para mayor claridad en la tendencia, no se muestran los datos de 200 pedidos.

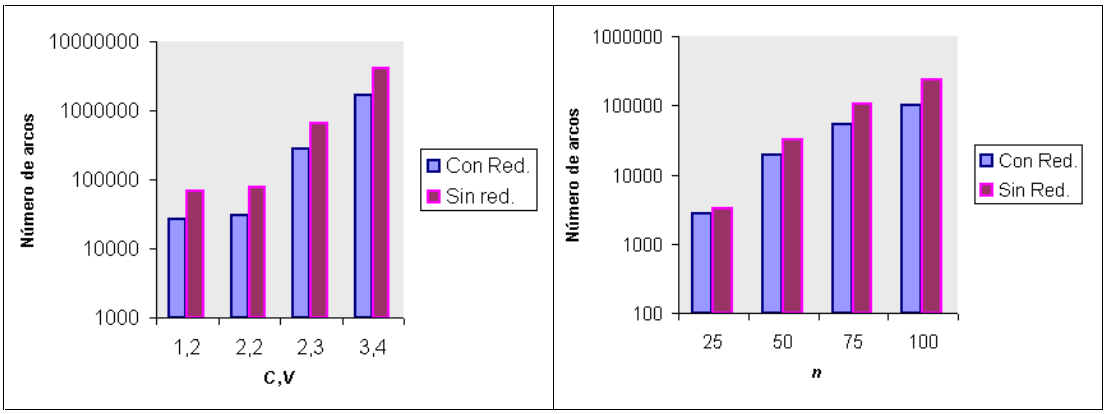


Figura 5.12: Número de arcos con y sin reducción

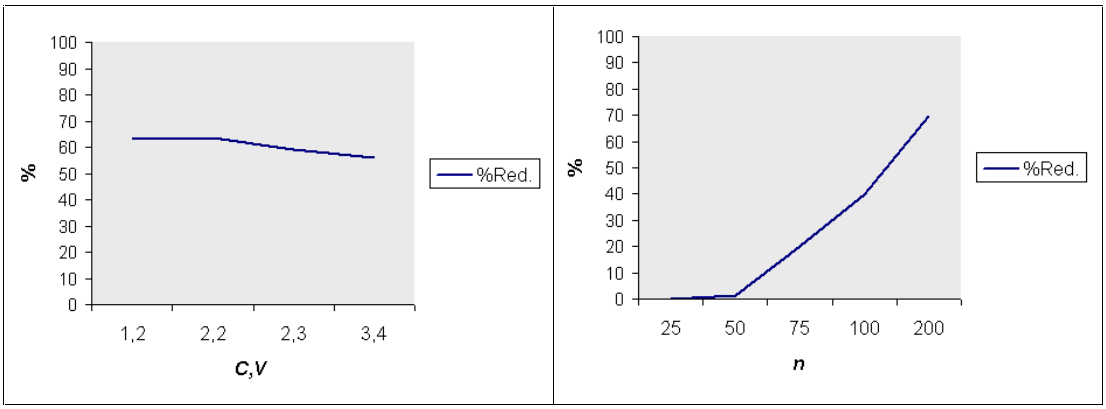


Figura 5.13: Porcentaje de reducción del número de arcos

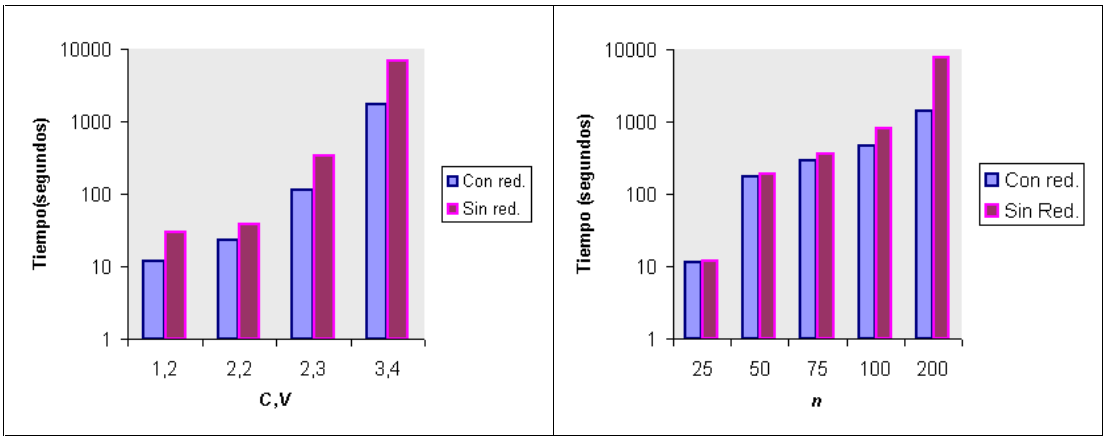


Figura 5.14: Tiempo de ejecución con y sin reducción

La figura 5.13 refleja la importancia de la regla de reducción de arcos. Como se observa en la figura, los porcentajes de reducción son independientes del número de recursos del problema, cayendo siempre en el intervalo en el intervalo [58%,61%]. Sin embargo, respecto al número de pedidos, los porcentajes de reducción experimentan un incremento notable con el aumento del mismo, siendo muy bajos con un número pequeño de pedidos. Este resultado se debe a que los tamaños de problemas en una misma batería poseen índices medios de solapamiento similares. Ello significa que mientras más pedidos existen, mayor es el horizonte temporal, y por tanto, con mayor probabilidad pueden darse situaciones en las que sea aplicable la regla de reducción.

Con respecto al grado de solapamiento, la figura 5.15 indica la tendencia de los porcentajes de reducción en el número de arcos respecto a las baterías 1, 3 y 4.

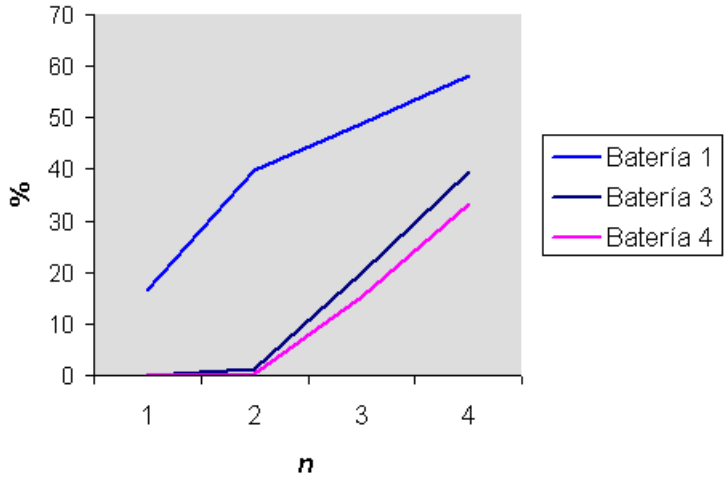


Figura 5.15: % reducción de número de arcos por batería

Puede observarse en la figura 5.15 que cuando aumenta el grado de solapamiento (ver tablas 5.5 y 5.6 y figura 5.8), la influencia de la regla de reducción es menor.

5.5.2.3. Comportamiento del método respecto al grado de simultaneidad

En esta sección se analizan los resultados obtenidos de la aplicación del método exacto con regla de reducción sobre las cuatro baterías de problemas. Los tamaños considerados fueron 25, 50, 75 y 100 pedidos. Las diferencia entre las baterías está en el grado de simultaneidad de los pedidos (ver tablas 5.5 y 5.6 y figura 5.8).

Para una mayor claridad, se muestran los datos de forma gráfica y agregados respecto a n y (C,V).

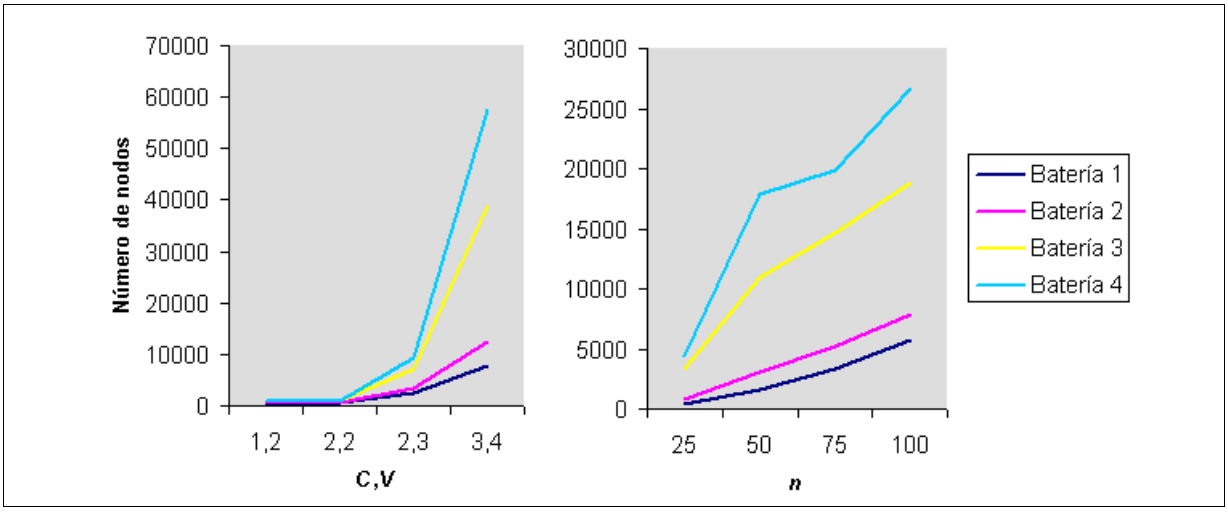


Figura 5.16: Número medio de nodos

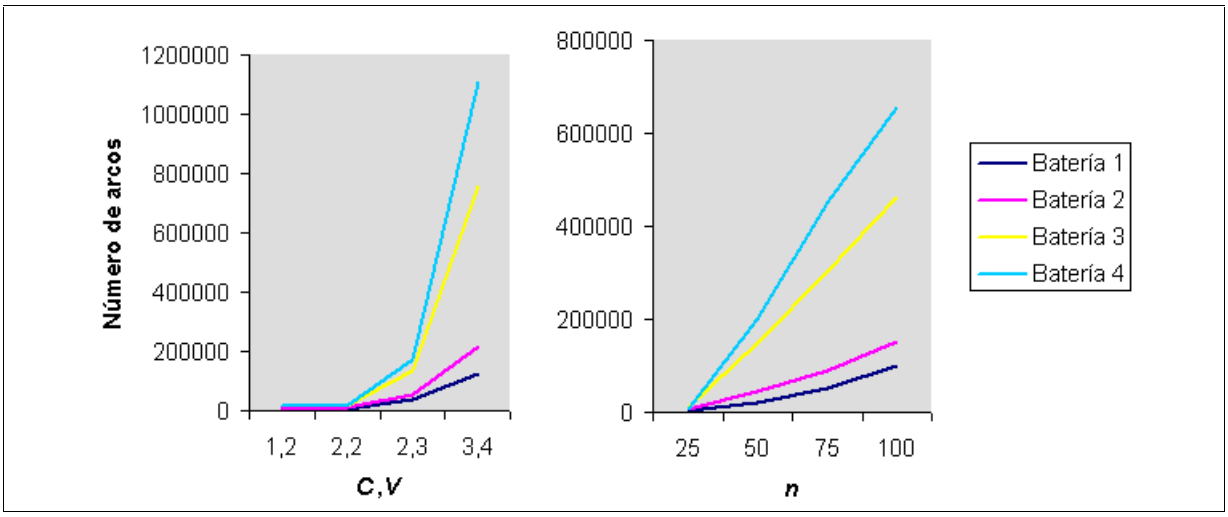


Figura 5.17: Número medio de arcos

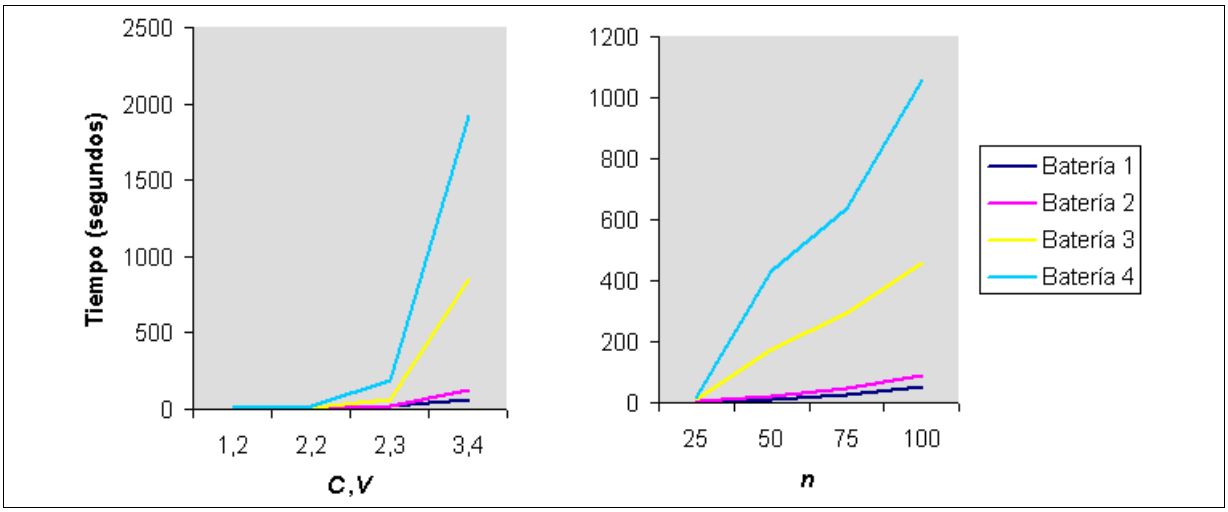


Figura 5.18: Tiempo medio de ejecución

Las gráficas muestran un comportamiento lineal del incremento del número de nodos, arcos y tiempo de ejecución respecto al incremento del grado de simultaneidad de la fase de distribución. Esto se ve de forma clara cotejando la gráfica 5.8 con las gráficas anteriores. Las baterías 3 y 4 experimentan un sensible incremento del número de nodos y arcos respecto a las baterías 1 y 2. Esto ocurre igualmente en el grado medio de simultaneidad en la fase de distribución. Esta fase es la que impone la tendencia en el problema, pues viene a ser un 80% de la actividad del pedido.

5.5.3. Resultados del procedimiento GRASP

En las ejecuciones realizadas con el método GRASP se utilizaron tres valores del parámetro α. Recordemos que α medía el grado de aceptación en la elección de los pedidos de la lista LRC. El número de iteraciones del método para todas las ejecuciones fue de 1000. La tabla 5.14 indica los promedios de error y el número

medio de iteraciones hasta la mejor solución. En todos los datos se hace media sobre las 10 instancias resueltas de cada tamaño. Los promedios de error agregados en función a C, V y n se presentan en la tabla 5.15. Finalmente, la tabla 5.16 muestra los tiempos de computación. La batería utilizada fue la Batería 1 de problemas.

		$\alpha = 0.7$		$\alpha = 0.8$		$\alpha = 0.9$	
(C,V)	***n***	**Error %**	**Nº iteraciones**	**Error %**	**Nº iteraciones**	**Error %**	**Nº iteraciones**
(1,2)	25	0.42	6.3	0.42	3.3	1.06	2.7
	50	0.07	152.6	0.38	49.7	1.66	10.2
	75	0.52	178.4	0.37	169.1	1.34	71.2
	100	1.23	473.2	0.98	441.3	2.19	233.1
	200	3.50	542.2	2.63	459.7	3.03	456.4
(2,2)	25	0.00	7.8	0.00	5.9	1.98	4.7
	50	0.10	249.0	0.63	47.5	1.43	27.8
	75	0.84	541.8	0.63	309.1	2.0	13.1
	100	1.46	497.5	1.27	446.9	2.12	391.9
	200	4.44	385.1	4.13	602.40	3.48	629.0
(2,3)	25	0.71	130.0	1.44	28.6	3.41	2.8
	50	3.21	317.7	3.86	351.0	4.95	33.6
	75	4.33	371.7	4.39	321.9	5.63	105.4
	100	6.14	563.8	6.14	448.4	7.13	229.5
	200	8.07	451.2	7.74	403.1	8.32	488.0
(3,4)	25	0.83	192.9	1.64	52.8	4.45	2.1
	50	4.97	554.8	5.92	175.0	7.70	99.9
	75	6.64	494.7	6.27	334.2	7.67	247.2
	100	8.75	460.6	8.72	625.9	9.64	297.2
	200	10.87	501.4	10.61	505.1	10.72	423.1

Tabla 5.14: Resultados del GRASP

	(C,V)						***n***					
α	(1,2)	(2,2)	(2,3)	(3,4)	Promedio	α	25	50	75	100	200	Promedio
0.7	1.15	1.37	4.49	6.41	3.35	0.7	0.49	2.09	3.08	4.39	6.72	3.35
0.8	0.95	1.33	4.71	6.63	3.40	0.8	0.87	2.70	2.91	4.28	6.28	3.40
0.9	1.85	2.20	5.88	8.04	4.49	0.9	2.72	3.93	4.16	5.27	6.39	4.49

Tabla 5.15: Promedios de error para (C,V) y n

(C,V)	*n*	Tiempo GRASP	Método Exacto
(1,2)	25	0.7	0.4
	50	1.5	2.8
	75	2.7	5.4
	100	4.2	8.5
	200	8.7	22.6
(2,2)	25	1.2	0.5
	50	2.4	2.9
	75	2.7	10.7
	100	3.9	22.8
	200	8.4	92.2
(2,3)	25	1.2	2.5
	50	2.1	15.8
	75	3.0	31.6
	100	4.8	52.9
	200	9.3	137.6
(3,4)	25	1.1	8.1
	50	2.4	64.6
	75	3.0	145.4
	100	4.5	281.6
	200	10.5	879.3

Table 5.16: Tiempos medios para (*C*,V) y *n*

Comentarios:

- De los tres valores considerados en el experimento para α, α=0.7 proporciona los mejores resultados, aunque los resultados para α=0.8 son similares. Sin embargo, cuando permitimos una menor elección aleatoria, α = 0.9, el método muestra un peor comportamiento. Aunque no mostrados, con valores de α inferiores a 0.7 el método presenta peor comportamiento.

- Es claro que GRASP presenta un mejor comportamiento cuando el número de vehículos y la capacidad de la planta es menor o igual a 2. Los peores resultados fueron para (*C*,*V*)=(3,4), donde el espacio de soluciones aumenta. Con respecto a *n*, los resultados lógicamente empeoran conforme incrementamos su valor sobre todo teniendo en cuenta que se ha mantenido el número de iteraciones para todos los problemas.

- Con referencia al número de soluciones óptimas, el algoritmo encontró el óptimo en 56 de los 200 problemas con α = 0.7.

- En general, los resultados proporcionados por el método GRASP se muestran satisfactorios para un sistema con un número reducido de recursos, (*C*,*V*) = (1,2) o (2,2). Para estos valores el error siempre estuvo por debajo del 5%. Cuando aumentamos los recursos el error aumenta sensiblemente a partir de 50 pedidos. La siguiente figura presenta gráficamente los errores medios.

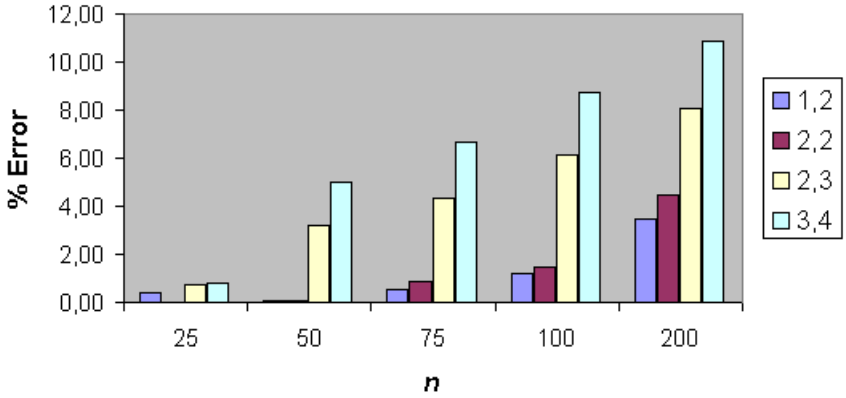


Figura 5.19: Error medio del procedimiento GRASP

- Los tiempos de ejecución son siempre mejores respecto al método de solución exacta, destacando la independencia del tiempo de ejecución respecto al número de recursos del problema.

5.5.4. Resultados del algoritmo de Búsqueda Tabú

El algoritmo de búsqueda tabú fue implementado con objeto de mejorar los resultados proporcionados por el algoritmo GRASP. Los experimentos realizados para la búsqueda tabú están divididos en tres apartados:

- En un primer apartado se realiza un análisis de sensibilidad de los parámetros fundamentales del método. Los experimentos para este análisis fueron realizados con la Batería 1 de problemas. (Sección 5.5.4.1)
- En el segundo apartado se comparan los métodos GRASP y Búsqueda Tabú. (Sección 5.5.4.2)
- En el tercer apartado, y una vez obtenida la configuración idónea de los parámetros del método, se analiza la eficiencia del método respecto al grado de simultaneidad de los pedidos. Para ello se comparan los resultados obtenidos en la resolución de las baterías 2, 3 y 4. (Sección 5.5.4.3)

5.5.4.1. Análisis de sensibilidad del algoritmo

El análisis de sensibilidad se enfoca respecto a tres parámetros del método:

- Tipo de solución inicial
- Tamaño de la lista tabú
- Número de iteraciones

Todas las tablas de resultados ofrecen, respecto a los valores de *C*, *V* y *n*, los datos medios de error y el porcentaje de óptimos alcanzados.

En los resultados correspondientes a cualquier parámetro, se ha tomado promedio de los resultados obtenidos de todas las ejecuciones realizadas con cada uno de los valores del resto de los parámetros.

En primer lugar, se efectúa una comparación de los valores obtenidos según la solución inicial empleada. Los tipos empleados, como ya se ha comentado, fueron soluciones iniciales aleatorias, soluciones obtenidas con un método *greedy*(avaricioso) y soluciones obtenidas con la aplicación del algoritmo GRASP anteriormente presentado, para el caso de una sola iteración y $\alpha = 0.8$.

De forma análoga, se presentan en segundo lugar los resultados respecto al tamaño de la lista tabú. Los tamaños de lista considerados fueron el número de pedidos pertenecientes a la solución inicial dividido por 3 y dividido por 2.

Respecto al número de iteraciones se analizaron 1000 y 5000 iteraciones con objeto de observar la disminución del error medio. También se muestran los tiempos empleados para esos números de iteraciones para una configuración dada.

- **Resultados según solución inicial**

	Aleatoria		Greedy		GRASP	
(*C*,*V*)	**% error**	**% óptimos**	**% error**	**% óptimos**	**% error**	**% óptimos**
(1,2)	1.53	38.5	0.91	47	0.83	49
(2,2)	1.05	45	1.13	42.5	1.13	50
(2,3)	1.16	35	0.99	36	0.90	38
(3,4)	0.84	36.5	0.64	43.5	0.55	44.5

Tabla 5.17: Resultados según solución inicial agregados por (*C*,*V*)

	Aleatoria		Greedy		GRASP	
n	**% error**	**% óptimos**	**% error**	**% óptimos**	**% error**	**% óptimos**
25	0.38	88.8	0.09	92.5	0.07	95
50	0.69	51.9	0.70	43.8	0.55	59.4
75	0.65	32.5	0.31	51.3	0.42	48.1
100	0.98	20	0.83	23.1	0.91	24.4
200	1.89	0.6	1.74	0.6	1.46	0

Tabla 5.18: Resultados según solución inicial agregados por *n*

Los resultados medios respecto a todas las ejecuciones son los siguientes:

	Aleatoria		Greedy		GRASP	
	% error	% óptimos	% error	% óptimos	% error	% óptimos
TOTAL	1.14	38.75	0.92	42.25	0.85	45.38

Tabla 5.19: Resumen de resultados según solución inicial

- **Resultados según el tamaño de la lista tabú**

	Nº pedidos solución/3		Nº pedidos solución/2	
(*C*,*V*)	% error	% óptimos	% error	% óptimos
(1,2)	1.22	41	0.96	48.7
(2,2)	1.17	43.7	1.04	48
(2,3)	1.05	35.3	0.98	37.3
(3,4)	0.70	40.7	0.65	42.3

Tabla 5.20: Resultados según tamaño de la lista tabú agregados por (*C*,*V*)

	Nº pedidos solución/3		Nº pedidos solución/2	
N	% error	% óptimos	% error	% óptimos
25	0.17	92.9	0.19	91.3
50	0.75	47.1	0.54	56.3
75	0.49	41.3	0.43	46.7
100	1.04	19.2	0.77	25.8
200	1.68	0.4	1.71	0.4

Tabla 5.21: Resultados según tamaño de la lista tabú agregados por *n*

	Nº pedidos solución/3		Nº pedidos solución/2	
	% error	% óptimos	% error	% óptimos
TOTAL	1.03	40.17	0.91	44.08

Tabla 5.22: Resumen de resultados según tamaño de la lista tabú

- **Resultados según el nº de iteraciones**

	1000		5000	
(*C*,*V*)	% error	% óptimos	% error	% óptimos
(1,2)	1.18	43.3	1	46.6
(2,2)	1.08	45.3	1.13	46.3
(2,3)	1	37	1.03	35.7
(3,4)	0.71	41	0.64	42

Tabla 5.23: Resultados según el nº de iteraciones agregados por (*C*,*V*)

	1000		5000	
n	**% error**	**% óptimos**	**% error**	**% óptimos**
25	0.16	92.5	0.20	91.7
50	0.67	49.6	0.62	53.8
75	0.48	43.8	0.44	44.2
100	0.93	22.1	0.88	22.9
200	1.73	0.4	1.66	0.4

Tabla 5.24: Resultados según el nº de iteraciones agregados por *n*

	1000		5000	
	% error	**% óptimos**	**% error**	**% óptimos**
TOTAL	0.99	41.67	0.95	42.58

Tabla 5.25: Resumen de resultados según el nº de iteraciones

La siguiente tabla presenta los tiempos medios de ejecución, expresados en segundos, para el algoritmo de búsqueda tabú con solución inicial GRASP, para 1000 y 5000 iteraciones.

n	***C***	***V***	***BT (1000)***	***BT (5000)***	***Método Exacto***
25	1	2	1.5	7.5	0.4
25	2	2	1.5	8.8	0.5
25	2	3	2	10	2.5
25	3	4	2.2	11.3	8.1
50	1	2	2	10.7	2.8
50	2	2	2.2	11.8	2.9
50	2	3	3	14.5	15.8
50	3	4	3.8	17	64.6
75	1	2	3	14.5	5.4
75	2	2	3.4	16.6	10.7
75	2	3	4	20.8	31.6
75	3	4	5	24.6	145.4
100	1	2	4	19	8.5
100	2	2	4.3	20.1	22.8
100	2	3	5.5	26	52.9
100	3	4	6.2	33.1	281.6
200	1	2	6.2	31.9	22.6
200	2	2	7.2	35.2	92.2
200	2	3	9.5	48	137.6
200	3	4	12.4	61	879.3

Tabla 5.26: Tiempos del algoritmo de búsqueda tabú

Comentarios

- El algoritmo de búsqueda tabú se mostró consistente respecto a los parámetros estudiados, presentando resultados homogéneos en todos los casos.

- El parámetro más relevante resultó ser la solución inicial empleada, con unos resultados previsibles. El comienzo del algoritmo con soluciones aleatorias presentó los peores resultados, siendo las soluciones iniciales GRASP las más efectivas.

- Los mejores resultados se obtuvieron para un tamaño de lista tabú igual a la mitad del número de pedidos en la solución inicial. Desviaciones sobre este valor se mostraron menos efectivas. Al tomarse un tamaño igual a un tercio de la solución inicial, el error aumenta un 0.10 %.

- Un aspecto importante es el estudio del número de iteraciones. Los resultados muestran que la mejoría de los resultados al aumentar de 1000 a 5000 iteraciones es mínima, por lo que puede concluirse que 1000 iteraciones es suficiente para un buen comportamiento del método. Aunque no presentes, también se estudiaron otros valores para el número de iteraciones. Para el caso de 500, el error fue sensiblemente peor que para 1000.

- Los tiempos de computación son similares a los tiempos obtenidos con el método GRASP, mejorando notablemente los tiempos del método exacto.

5.5.4.2. Comparación con el algoritmo GRASP

Para la comparativa entre los dos algoritmos, los parámetros de la búsqueda tabú tomaron los siguientes valores:

- Solución inicial obtenida por el método GRASP (una iteración)

- Tamaño de la lista tabú = Nº pedidos en la solución / 2

- Nº iteraciones = 1000

Las figuras 5.20 y 5.21 muestran el error medio en función de n y (C,V), respectivamente. La figura 5.22 presenta los tiempos medios de ejecución de los dos algoritmos.

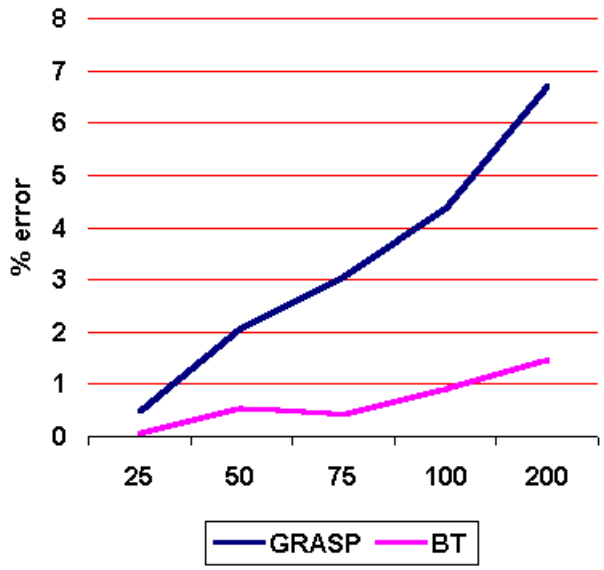


Figura 5.20: Error medio de GRASP y BT respecto a n

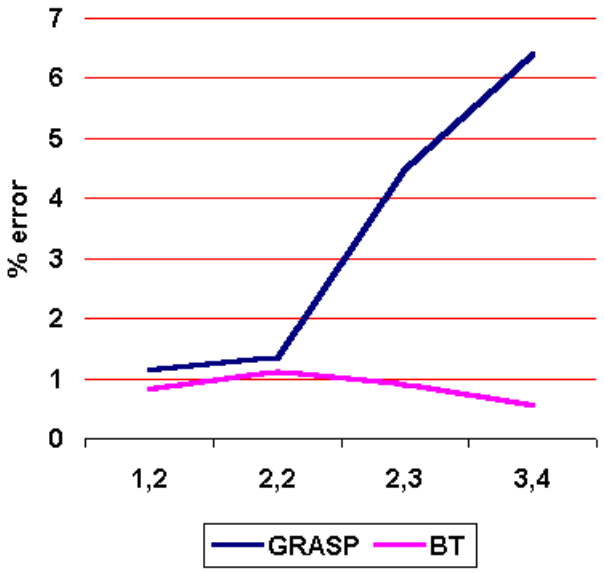


Figura 5.21: Error medio de GRASP y BT respecto a (C,V)

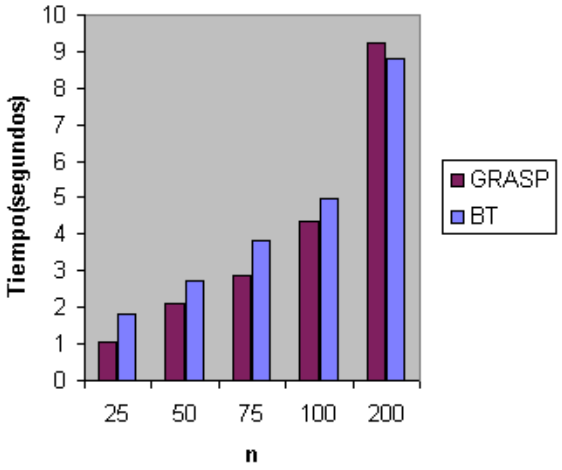


Figura 5.22: Tiempos medios de GRASP y BT

Comentarios

- Los resultados del algoritmo de búsqueda tabú mejoran notablemente los resultados proporcionados por el algoritmo GRASP.
- Los tiempos son similares en ambos métodos.
- La característica más importante de la búsqueda tabú, que lo diferencia del método GRASP, es que no incrementa el error cuando se incrementan los recursos (C,V) del problema.

5.5.4.3. Comportamiento del algoritmo respecto al grado de simultaneidad

En esta tercera etapa del estudio de la búsqueda tabú se presentan los resultados obtenidos sobre tres baterías diferentes de problemas. Las baterías consideradas han sido la batería 2, 3 y 4. Los grados de simultaneidad de cada batería, tomando promedio respecto a todos los tamaños de problemas, son los siguientes:

	$L^P{}_m$	$L^D{}_m$	L^P	L^D
Batería 2	2.2	9.1	6.3	16.0
Batería 3	2.6	13.0	6.5	22.0
Batería 4	3.2	15.1	7.5	24.6

Tabla 5.27: Grados de simultaneidad

Para la configuración de los parámetros del algoritmo se ha tomado en consideración el primer apartado del análisis. Según ello, el mejor comportamiento del método se obtuvo para una solución inicial obtenida mediante un procedimiento GRASP. El tamaño de la búsqueda tabú se consideró el número de elementos en la solución inicial partido por dos. En todas las ejecuciones se consideran 1000 iteraciones.

A continuación se presentan las tablas con los resultados de todas las ejecuciones. En todos los casos se realiza promedio sobre las 10 instancias de cada tamaño de problema. Las tablas recogen el error medio, el % de pedidos servidos, los tiempos de ejecución y el número de soluciones óptimas (respecto a las 10 instancias en cada caso).

		Error (%)			Pedidos servidos (%)		
(*C*,*V*)	***n***	**Batería 2**	**Batería 3**	**Batería 4**	**Batería 2**	**Batería 3**	**Batería 4**
(1,2)	**25**	0.00	1.20	1.96	33.2	16.0	10.4
	50	0.69	1.95	2.43	25.4	14.8	13.0
	75	0.38	2.74	3.09	28.3	14.9	13.1
	100	0.49	3.63	4.82	27.9	14.9	13.6
	200	1.63	5.18	4.50	23.6	14.8	13.4
(2,2)	**25**	0.22	1.90	1.10	35.2	16.0	10.4
	50	0.72	1.90	2.75	26.2	14.8	13.0
	75	0.69	3.27	3.70	29.2	15.2	13.3
	100	1.06	4.00	6.09	28.6	14.9	13.7
	200	3.26	6.07	5.26	25.3	15.2	13.8
(2,3)	**25**	0.04	1.58	0.00	48.0	22.8	14.4
	50	0.51	0.71	2.93	37.6	21.6	18.8
	75	0.66	2.98	2.79	42.0	21.9	19.1
	100	1.00	2.77	3.83	40.4	21.6	19.9
	200	1.97	4.14	4.81	36.2	22.1	20.0
(3,4)	**25**	0.00	0.89	0.00	61.6	28.4	18.4
	50	0.32	2.02	1.68	49.4	28.0	24.6
	75	0.45	2.15	1.42	53.5	28.1	24.4
	100	1.20	2.63	2.51	51.8	28.0	25.5
	200	1.16	3.44	4.28	47.1	28.5	26.5

Tabla 5.28: Porcentajes de error y de pedidos servidos

		Nº soluciones óptimas			Tiempos medios (segundos)		
(*C*,*V*)	***n***	**Batería 2**	**Batería 3**	**Batería 4**	**Batería 2**	**Batería 3**	**Batería 4**
(1,2)	**25**	10	8	7	1.7	1.1	1
	50	7	8	6	2.4	1.7	1.6
	75	6	3	0	3.0	2.5	2.3
	100	3	2	0	3.9	2.9	2.6
	200	0	0	0	6.5	4.9	4.6
(2,2)	**25**	9	7	8	1.7	1.4	1.2
	50	7	4	3	2.3	1.8	1.6
	75	6	2	2	3.4	2.2	2.1
	100	2	1	1	4.1	2.8	2.8
	200	0	0	0	6.9	5.1	5.1
(2,3)	**25**	9	6	10	1.9	1.2	1.1
	50	6	6	4	2.9	2.2	2.2
	75	4	1	2	4.1	3.1	3
	100	1	2	0	5.2	3.9	3.6
	200	0	0	0	9.8	7.3	6.9
(3,4)	**25**	10	8	10	2.1	1.6	1.5
	50	7	5	3	3.1	2.7	2.8
	75	5	2	2	5.0	3.8	3.5
	100	1	0	0	6.5	4.9	4.9
	200	0	0	0	12.6	9.2	8.9

Tabla 5.29: Número de óptimos y tiempos medios

Agregando la información por (*C*,*V*) y *n*, resultan los siguientes valores:

(*C*,*V*)	**(1,2)**	**(2,2)**	**(2,3)**	**(3,4)**	***n***	**25**	**50**	**75**	**100**	**200**
Batería 2	0.64	1.19	0.83	0.63	Batería 2	0.06	0.56	0.55	0.94	2.01
Batería 3	2.94	3.43	2.44	2.23	Batería 3	1.39	1.65	2.79	3.26	4.71
Batería 4	3.36	3.78	2.87	1.98	Batería 4	0.77	2.45	2.75	4.31	4.71

Tabla 5.30: Error medio agregado

(*C*,*V*)	**(1,2)**	**(2,2)**	**(2,3)**	**(3,4)**	***n***	**25**	**50**	**75**	**100**	**200**
Batería 2	26	24	20	23	Batería 2	38	27	18	10	0
Batería 3	21	14	15	15	Batería 3	29	23	8	5	0
Batería 4	13	14	16	15	Batería 4	35	16	6	1	0

Tabla 5.31: Nº medio de óptimos agregado

(*C*,*V*)	**(1,2)**	**(2,2)**	**(2,3)**	**(3,4)**	***N***	**25**	**50**	**75**	**100**	**200**
Batería 2	3.5	3.6	4.8	5.8	Batería 2	1.8	2.6	3.9	4.9	8.9
Batería 3	2.6	2.6	3.5	4.4	Batería 3	1.3	2.1	2.9	3.6	6.6
Batería 4	2.4	2.5	3.3	4.3	Batería 4	1.2	2.0	2.7	3.4	6.3

Tabla 5.32: Tiempos medios agregados

Las siguientes figuras reflejan los valores de las tablas:

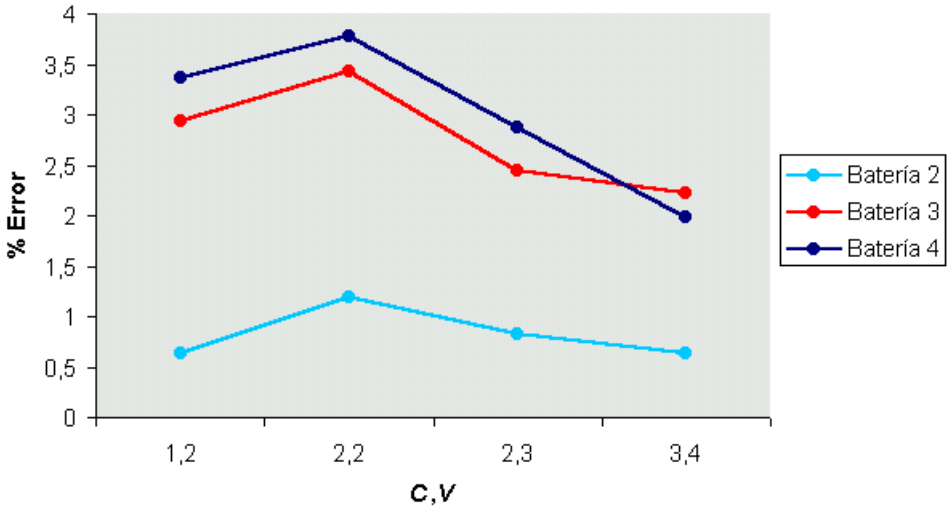


Figura 5.23: Error medio respecto a (C,V)

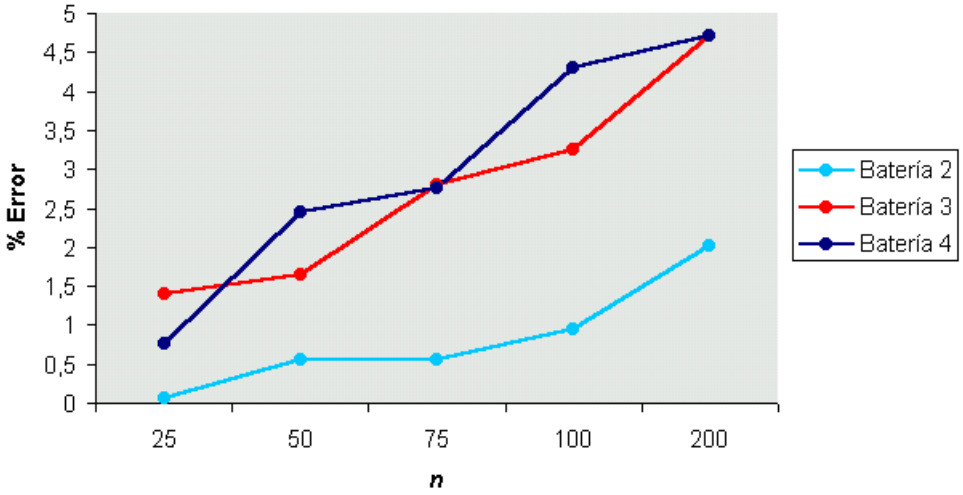


Figura 5.24: Error medio respecto a n

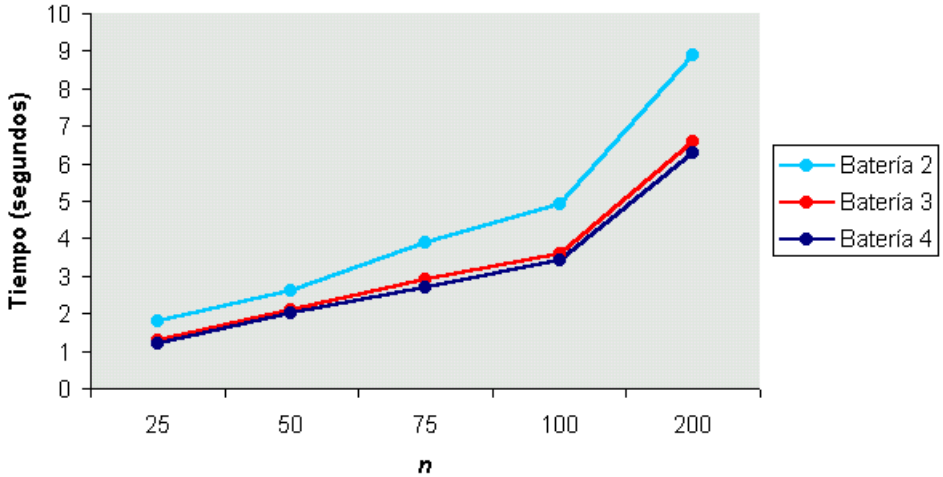


Figura 5.25: Tiempo medio respecto a n

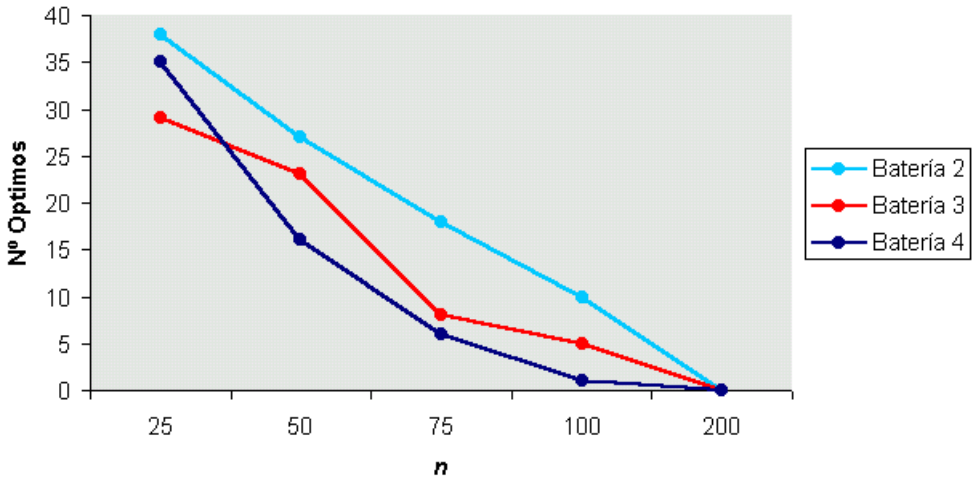


Figura 5.26: Nº óptimos respecto a *n*

Comentarios

A la vista de los resultados obtenidos pueden extraerse las siguientes conclusiones:

- El error medio obtenido experimenta una subida mínima con respecto al grado de solapamiento de los pedidos, considerándose estable a partir de un cierto nivel de solapamiento (figuras 5.25 y 5.26).

- Se corrobora que el crecimiento en el número de recursos no aumenta el error medio proporcionado por el algoritmo. Éste es el resultado más importante y satisfactorio que podemos extraer del análisis del Escenario IA, puesto que la aplicación del método exacto se vuelve intratable con el aumento de recursos.

- El error medio en función del número de pedidos aumenta de forma bastante lineal. Obviamente, la pendiente del error aumenta más al pasar de 100 a 200 pedidos, pues la escala respecto al número de pedidos se ha comprimido (figura 5.26).

- Los tiempos del algoritmo se reducen con el aumento en el grado de solapamiento. La razón radica en la disminución del número de pedidos candidatos a formar parte de la solución, en cada iteración del método, lo cual repercute en una disminución de la vecindad. Cuanto menor es la dispersión de los pedidos en el horizonte temporal, más fácil es que un pedido viole los recursos disponibles al ser introducido en la solución. Por ello, con una menor dispersión disminuye la vecindad de una solución.

- El número de óptimos disminuye de forma lineal en las tres baterías con respecto al aumento en el número de pedidos, encontrándose en sintonía con los datos del error medio. Además de la dificultad obvia de obtener

soluciones óptimas cuando aumenta el número de pedidos, es importante resaltar que fueron 1000 iteraciones las utilizadas en todos los problemas, lo cual repercute en una disminución del porcentaje de exploración de soluciones cuando aumenta el tamaño del problema.

Capítulo 6

Escenario IB: Número ilimitado de vehículos

Índice

6.1. INTRODUCCIÓN

Un número ilimitado de vehículos no significa un número infinito, sino simplemente la garantía de que siempre haya un vehículo disponible. Dicho de otro modo, si el mayor grado de simultaneidad L^D en la fase de distribución es m, entonces ese valor m es una cota inferior del número de vehículos requeridos. Siempre que el número de vehículos sea mayor o igual que m estaremos en el caso de un número ilimitado de vehículos.

Para presentar una variante en uno de los factores de estudio del problema PDP, en esta sección se supone que el vehículo interviene en la fase de producción del pedido. La única variación que supone esto, queda expresada en la formulación del problema. La metodología de resolución no depende de este factor.

El objetivo de este escenario es el estudio del problema PDP teniendo en cuenta únicamente uno de los dos recursos limitados, es decir, únicamente la planta de producción. Además, asignando el mismo valor a todos los pedidos, se plantea un segundo objetivo en el problema que trate de minimizar el número de vehículos necesarios para la distribución del número máximo de pedidos calculado.

El objetivo de minimizar el número de vehículos está siempre presente a la hora de asignar vehículos a un conjunto de pedidos seleccionados. Sin embargo, en el caso de asignar un mismo valor a todos los pedidos, el objetivo de maximizar el valor de los pedidos se convierte en maximizar el número de pedidos servidos. Este criterio puede llevar con frecuencia a la aparición de soluciones óptimas alternativas, todas ellas sirviendo el número máximo de pedidos, pero utilizando un número de vehículos que puede ser diferente. El problema tratará entonces de encontrar aquélla que utilice el menor número de vehículos.

El capítulo está organizado del siguiente modo:

- En la sección 6.2 se plantea el problema general con objeto de maximizar el valor de los pedidos servidos.

- En la sección 6.3 se plantea el caso en el que el valor de los pedidos es idéntico, utilizando como criterio la minimización del número de vehículos para atender el número máximo de pedidos a ser servidos. Para este caso se presenta un método de solución exacta y un método heurístico eficiente basado en un esquema de acotación y ramificación, cuyos resultados se presentan en el apartado de experimentación.

6.2. MAXIMIZAR EL VALOR DE LOS PEDIDOS SERVIDOS

La figura 6.1 muestra el diagrama temporal de un conjunto de pedidos. Para este ejemplo L^P = 2, por tanto, con una capacidad en planta de 2 unidades podrían procesarse todos los pedidos. Sin embargo, si $C < 2$ tendríamos que seleccionar un subconjunto de pedidos para ser servidos.

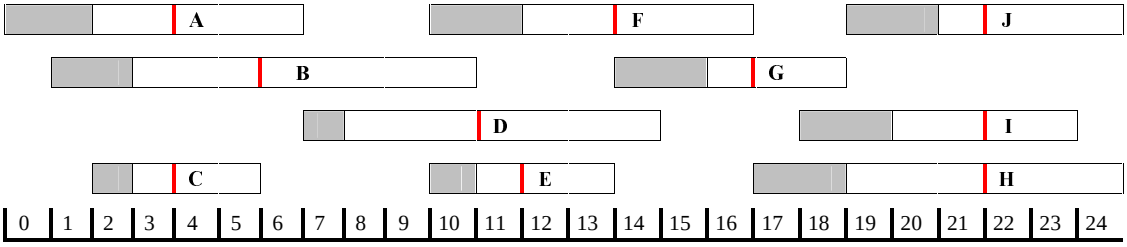


Figura 6.1: Ejemplo de plan de producción

El problema PDP en este caso particular se hace equivalente a un problema de programación de trabajos con tiempos fijos(FSP) [Kroon et al, 1995][Arkin y Silverberg, 1987][Fishetti et al, 1992][Gabrel, 1995] descrito en el Capítulo 3. También es equivalente a la maximización, sobre máquinas en paralelo, del peso de los trabajos finalizados justo en su fecha de entrega [Hiraishi et al, 2002][Lann y Mosheiov, 2002], suponiendo fechas de llegada al sistema. En concreto, sería equivalente al problema FSP con una sola clase de máquinas y trabajos. La mayoría de los autores muestran que este problema es resuelto en tiempo polinomial por un algoritmo de flujo a coste mínimo en un grafo. A continuación, describimos la construcción de un nuevo grafo para la resolución del problema.

Sea n el número de pedidos recibidos. Asumiendo la formulación ya establecida para el Escenario I del problema PDP (Capítulo 4), el modelo correspondiente al caso de vehículos ilimitados es el siguiente:

$$Max \quad \sum_{i=1}^{n} w_i p_i$$

$$s.a.$$

$$\sum_{i \in S(t)} p_i \leq C \qquad t = 0 \ldots T \quad (1)$$

$$p_i = \begin{cases} 1 & \text{si el pedido } i \text{ es servido} \\ 0 & \text{en otro caso} \end{cases}$$

La función objetivo es el valor total de los pedidos servidos. Las restricciones (1) imponen la capacidad limitada en la planta. Del modelo planteado en el Capítulo 4 se han eliminado las restricciones asociadas a la limitación en el número de vehículos disponibles.

Para resolver el problema proponemos un algoritmo de flujo, a partir de la idea propuesta por [Kroon et al, 1995] para el problema FSP, basado en un método de construcción que reduce el tamaño del grafo. La construcción del grafo $G(N,A)$ es la siguiente. Para cada pedido i se calcula Posterior(i) como el primer pedido cuyo proceso de fabricación comienza después de finalizar el proceso de fabricación de i. Si mantenemos los pedidos ordenados por s_i, Posterior(i)={k / k=Mínimo$_j$ ($s_j \geq l_i$) $\forall j$}. Por cada pedido asociamos un nodo en el grafo, N = {1,..., n}, más un nodo final n+1, que representa el posterior de todos los pedidos que no poseen pedidos posteriores. El nodo 1 actúa como nodo fuente del grafo.

Por cada pedido i, i = 1..n, existe un arco desde el nodo i hasta el nodo Posterior(i) con una unidad de flujo de capacidad y un coste igual a $-w_i$. Denominamos estos arcos como el conjunto A^P. Además, se añade un arco de coste cero y capacidad ilimitada desde el nodo i al nodo i+1, para i = 1...n, que incluimos en el conjunto A^M. Así, el conjunto de arcos del grafo es $A = A^P \cup A^M$. El flujo de entrada en el grafo es C, la capacidad de producción en la planta. El nodo sumidero n+1, será el nodo con déficit de C unidades de flujo. El grafo para el ejemplo de la figura 6.1 quedaría como sigue:

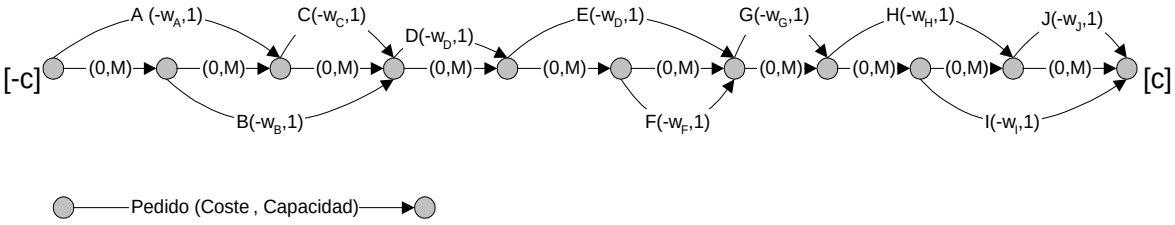


Figura 6.2: Ejemplo de grafo $G(N,A)$.(M = ∞)

En el grafo de la figura 6.2 está representado el valor ∞ con la notación M. Como notación, se representa en positivo el déficit de flujo y en negativo el origen de flujo.

Comparando el grafo propuesto por [Kroon et al, 1995] con el anterior, los resultados son los siguientes:

	Grafo [Kroon et al, 1995]	**Grafo propuesto**
Número de Nodos	2n	n+1
Número de Arcos	3n-1	2n

Tabla 6.1: Comparación de estrategias

En definitiva, el grafo propuesto disminuye el nº de nodos y el nº de arcos en n-1. Los pedidos seleccionados serían aquellos por cuyos arcos asociados (A^P) circula una unidad de flujo en la solución óptima del problema:

$$i : (i, \text{Posterior}(i)) \in \text{ ruta de coste mínimo.}$$

La solución óptima a este problema se puede obtener a través de diferentes algoritmos. El que hemos usado en este trabajo ha sido RELAX IV [Bertsekas y Tseng, 1988], en particular la implementación C++ disponible en [Frangioni, 2001].

Ilustremos el método propuesto para el ejemplo de la figura 6.1 para una capacidad de fabricación de C = 1 y los siguientes valores de los pedidos:

Pedidos	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
w_i	4	11	6	10	8	7	9	4	10	4

Tabla 6.2: Valores de los pedidos

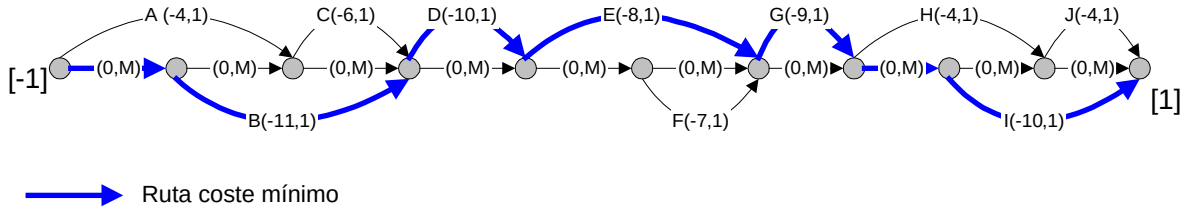


Figura 6.3: Flujo a coste mínimo

Los pedidos que forman parte de la ruta de coste mínimo hacen máximo el beneficio con un valor total de 48 unidades. Visto gráficamente:

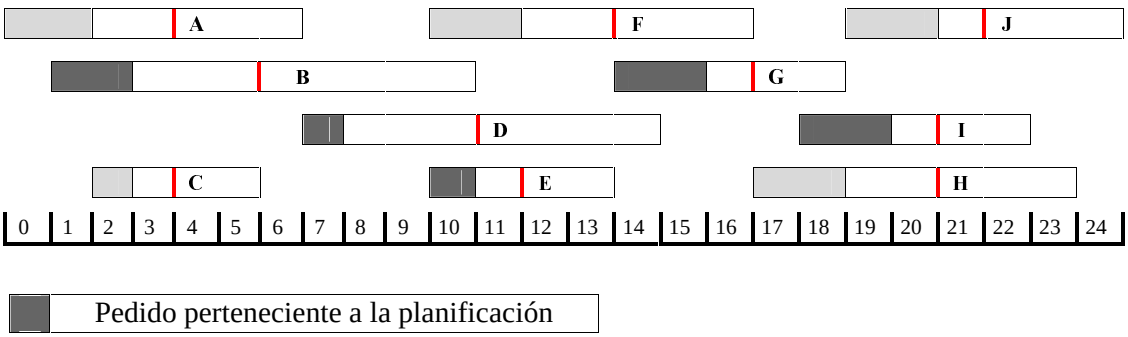


Figura 6.4: Diagrama resultado

Una vez que se conocen los pedidos que van a ser servidos, se deben asignar los vehículos de forma que el número de ellos sea mínimo. Esto puede ser modelado como sigue:

$$Min \sum_{j=1}^{m} y_j$$

$$\sum_{j=1}^{m} x_{ij} = p_i^* \qquad i = 1 \ldots n \quad (1)$$

$$x_{ij} + x_{kj} \leq 1 \qquad i = 1 \ldots n-1;\ j = 1 \ldots m;\ \{k > i : s_k < f_i\} \quad (2)$$

$$x_{ij} \leq y_j \qquad i = 1 \ldots n;\ j = 1 \ldots m \quad (3)$$

$$x_{ij} = \begin{cases} 1 & \text{si el pedido } i \text{ se asigna al vehiculo } j \\ 0 & \text{en otro caso} \end{cases}$$

$$y_j = \begin{cases} 1 & \text{si el vehiculo } j \text{ es usado} \\ 0 & \text{en otro caso} \end{cases}$$

Aunque el número de vehículos es desconocido, pero lo suficiente como para que siempre existan vehículos disponibles, en la formulación del problema se supone una cantidad de m vehículos; $m \geq L^D$.

La función objetivo minimiza el número de vehículos requerido. Las restricciones (1) imponen consistencia entre los valores p_i^* y las variables x_{ij} donde p_i^* corresponde a la solución óptima obtenida a través de la aplicación del algoritmo de flujo a coste mínimo descrito anteriormente. Las restricciones (2) garantizan que cuando a un pedido se le asigna un vehículo, el vehículo no puede asignarse a ningún otro pedido, hasta que el pedido no haya sido completado y el vehículo haya retornado a la planta. Finalmente, las restricciones (3) imponen consistencia entre las variables x_{ij} y la variables y_j.

Este problema puede ser resuelto por una heurística de tipo *greedy* (avariciosa)[Fischetti et al 1990], cuyo procedimiento está descrito en el Capítulo 3. El resultado de la aplicación del algoritmo sobre nuestro ejemplo sería de 3 vehículos. El vehículo 1 serviría {B, G}; el vehículo 2 serviría {D,I}; y el vehículo 3 serviría el pedido {E}.

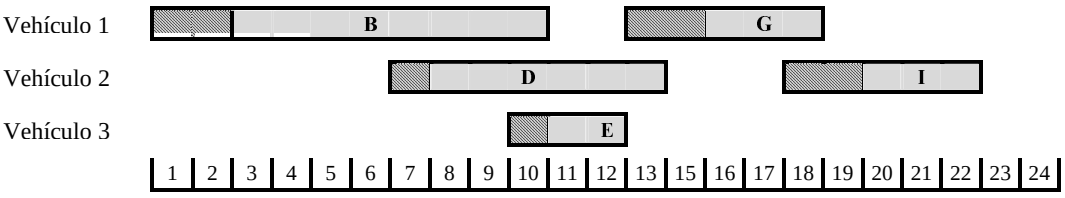


Figura 6.5: Asignación de pedidos a vehículos

Si hubiéramos supuesto que el vehículo no interviene en la fabricación, el resultado hubiera sido el siguiente:

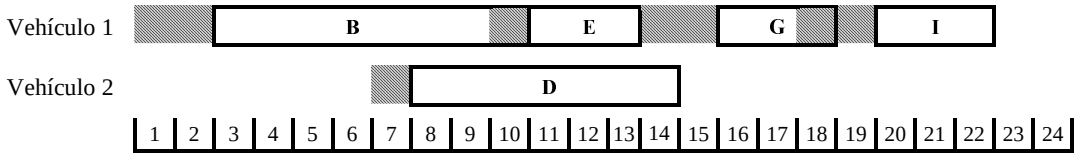


Figura 6.6: Asignación de pedidos a vehículos sin participación en producción

El número de vehículos sería en ese caso de 2. Los pedidos que se realizan son independientes de la participación o no del vehículo en el proceso de producción, pues dependen únicamente de la capacidad de fabricación en la planta. Con ello hacemos hincapié en el hecho de que la participación del vehículo no influye en la metodología de resolución.

6.3. MAXIMIZAR EL NÚMERO DE PEDIDOS

Este segundo enfoque considera la situación en la que el peso de los pedidos es desconocido o el mismo para todos los pedidos. Suponemos, por tanto, $w_i = 1\ \forall i$.

Usando el enfoque y método de resolución descritos en la sección anterior, se obtiene el máximo número de pedidos servidos, denominado p^*, que se computa como la solución óptima formada por p^* pedidos. Sin embargo, puede suceder a menudo que existan soluciones óptimas alternativas, todas ellas sirviendo p^* pedidos, pero usando un número variable de vehículos. Por ello, sería interesante seleccionar aquella solución que utiliza el menor número de vehículos. El modelo para esta situación sería el siguiente:

$$Min \sum_{j=1}^{m} y_j \quad (1)$$

sujeto a

$$x_{ij} \le y_j \qquad i = 1 \dots n;\ j = 1 \dots m$$

$$\sum_{j=1}^{m} x_{ij} = p_i \quad i = 1 \dots n$$

$$\sum_{i=1}^{n} p_i = p^* \quad (2)$$

$$x_{ij} + x_{kj} \le 1 \qquad i = 1 \dots n-1;\ \ j = 1 \dots m;\ \{k > i : s_k < f_j\}$$

$$\sum_{i \in S(t)} p_i \le C \qquad t = 0 \dots T$$

$$x_{ij} \in \{0,1\} \quad y_j \in \{0,1\} \quad p_i \in \{0,1\}$$

La función objetivo (1) minimiza el número de vehículos a utilizar. Además de las restricciones comunes al enfoque de la sección anterior, en el modelo se añade la restricción (2) que garantiza que los pedidos seleccionados para cualquier solución del modelo sean exactamente p^*.

6.3.1. Un método de solución exacta basado en grafos

La razón fundamental para desarrollar este enfoque basado en grafos es que sus resultados pueden útilmente guiar las conclusiones sobre la heurística descrita en la siguiente sección. El método que presentamos está basado en un efectivo método para el problema PDP con un número fijo de vehículos descrito en el Escenario IA. Este método construye un grafo *G* que colecciona todas las soluciones admisibles del problema a través de un simple proceso de evaluación de estados admisibles en la planificación de los pedidos. El camino más largo desde el nodo de salida de *G* hasta el nodo final es la solución óptima del problema. Sea *GrafoExacto(P,C,v)* la función que construye y resuelve el grafo *G* para un conjunto *P* de pedidos, con *C* unidades de capacidad y *v* vehículos. El valor que retorna será por lo tanto, el número máximo de pedidos que pueden ser procesados haciendo uso de esos recursos.

Comenzando desde el cálculo del número máximo de vehículos requerido *m*, el cual se obtiene como el grado máximo de simultaneidad, utilizamos un simple proceso repetitivo para encontrar el número mínimo de vehículos requerido para servir p^* pedidos. El número máximo de pedidos p^* se obtiene haciendo uso de la función GrafoExacto(*P*,*C*,*v*) cuando *v* es igual al mayor grado de simultaneidad. Tras calcular p^*, en cada iteración del algoritmo se reduce en uno el número de vehículos y se vuelve a calcular el grafo exacto. El proceso finaliza cuando el valor calculado sobre el grafo sea menor a p^*. El pseudocódigo de este procedimiento se muestra a continuación.

```
Procedimiento Solucion_Exacta (P={1…n}, C)
    m = máximo grado de simultaneidad(P)
    p* = GrafoExacto(P, C, m)
    Repetir
        v* = m
        m = m - 1
    Hasta que p* <> GrafoExacto(P, C, m)
    Devolver (v*)
Fin
```

Desafortunadamente, dado que el número de nodos y arcos del correspondiente grafo es de orden $O(n^{C+v})$, la aplicación práctica de este enfoque es reducida. Este hecho fuerza a considerar, para valores medios y altos de n y C, una heurística alternativa descrita en la siguiente sección.

6.3.2. Una heurística eficiente basada en un esquema de ramificación y acotación

La estrategia que sigue la siguiente heurística consiste en considerar sustituciones de pedidos de una solución original, manteniendo siempre un número p^* de pedidos servidos. La evaluación de tales sustituciones se lleva a cabo usando un esquema de ramificación y acotación. Lo que la diferencia de un método exacto es que no se explora el árbol completo.

Una solución admisible para el problema considerado en esta sección es cualquiera que sirve p^* pedidos. El algoritmo comienza con la solución obtenida usando el algoritmo de flujo a coste mínimo descrito en la sección 6.2, el cual proporciona la solución óptima respecto al número de pedidos seleccionados, es decir, proporciona el valor p^*. Sea S_o la solución original obtenida, es decir, el conjunto inicial de p^* pedidos seleccionados. Sea O su complementario, el conjunto de pedidos fuera de la solución. La solución S_o corresponde con el nodo raíz en el árbol. Sea $V(S_o)$ el número de vehículos requeridos por la solución inicial. Este valor es una cota superior del número óptimo de vehículos requerido.

En cada nodo j del árbol de exploración, un subconjunto específico de pedidos $K_j \subseteq O$ se introduce en la solución a través del procedimiento descrito en la sección 6.3.2.1. El primer nivel corresponde a conjunto de un solo pedido, $K_j = \{k\}$ por cada $k \in O$. Por tanto, existen $|O| = n\text{-}p^*$ nodos en el primer nivel (j=1… $n\text{-}p^*$). El segundo nivel contiene $(n\text{-}p^*)(n - p^* - 1)/2$ nodos, cada uno de ellos con un conjunto asociado $K=\{k_1, k_2\}$ $k_1<k_2$; $k_1, k_2 \in O$.

En general, el D-ésimo nivel posee $\binom{n-p^*}{D}$ nodos.

El árbol completo tiene en principio 2^{n-p^*} nodos. Sin embargo, el método heurístico no explorará todos los niveles sino únicamente un número específico de ellos, establecido como parámetro del método. Para ello, la exploración sigue un recorrido en anchura del árbol de exploración.

Cuando los pedidos pertenecientes a K_j son forzados a pertenecer a la solución, puede darse uno de los dos casos siguientes:

1. No existe una solución admisible con p^* pedidos: Entonces ese nodo puede ser podado. Además, el conjunto K_j se añade a una lista de conjuntos de pedidos

prohibidos, para que si en otro nodo del árbol el conjunto de pedidos a forzar en la solución contiene los pedidos de K_j, ese nodo puede ser también, con seguridad, podado. Esto acelera notablemente la exploración, sugiriendo la estrategia de búsqueda en anchura ya mencionada.

2. Se encuentra una solución admisible con p^* pedidos: Esta nueva solución es también óptima respecto al número de pedidos seleccionados. El número de vehículos requerido se computa usando el método de asignación avariciosa de vehículos [Fischetti et al, 1992], y se compara con la cota superior. Sea V_j ese número de vehículos. Si V_j es menor que la cota superior, significa que se ha encontrado una solución admisible de cardinalidad p^* que requiere un menor número de vehículos que la mejor solución encontrada hasta ese momento. Por tanto, se modifica la mejor solución V^* con el valor V_j, pasando a ser la nueva cota superior.

6.3.2.1. Introducir pedidos en la solución

Los pedidos de K_j deben ser introducidos en la solución para chequear si existe una nueva solución con p^* pedidos y, en ese caso, calcular el número de vehículos requeridos por esa solución. Para hacer esto, los pedidos de K_j son forzados a ser servidos añadiendo la siguiente restricción al modelo definido en la sección 6.2:
$p_i = 1 \quad \forall i \in K_j$

Por tanto, el problema que denotamos como $P(j)$ a resolver en cada nodo es el siguiente:

$$
\begin{aligned}
& Max \ \sum_{i=1}^{n} w_i p_i \\
& s.a. \\
& p_i = 1 \quad \forall i \in K_j \\
& \sum_{i \in S(t)} p_i \leq C \quad t = 0 \ldots T \\
& p_i \in \{0,1\}
\end{aligned}
$$

Aunque los coeficientes w_i en este escenario son igual a 1 para todos los pedidos, para el problema anterior se modifican en cada nodo para dar prioridad a los pedidos pertenecientes a la solución original S_o. Ello significa que en cada nodo necesitamos construir una solución que fuerce los pedidos de K_j pero que a la vez mantenga un número máximo de pedidos pertenecientes a la solución original S_o. De ese modo, siempre que un pedido de la solución original no pertenezca a la solución de un nodo, es porque ha sido intercambiado por un pedido de K_j.

Obviamente, el número de pedidos intercambiados será menor o igual a p^* - $|K_j|$, ya que no existe garantía de que todos los pedidos de K_j puedan ser insertados dando lugar a una solución admisible.

Por ello, para dar prioridad a los pedidos de S_o, para cada $i \in S_o$ se asigna un valor $w_i = M_i$, obtenido de acuerdo a la siguiente formula: $M_i = M \cdot |Q_i|$, con M un número grande y $Q_i = \{k : f_k \le s_i \,\|\, s_k \ge f_i\}$, es decir, Q_i corresponde con el conjunto de pedidos que no se solapan con el pedido i. Podríamos haber dado prioridad a esos pedidos si les hubiéramos simplemente asignado un valor M_i igual a M. Sin embargo, alcanzamos un mejor comportamiento si multiplicamos ese valor M por Q_i ya que de esta forma distinguimos los mejores pedidos de entre los pedidos de la solución inicial. Consideramos que un pedido es mejor que otro cuando se solapa con un número menor de pedidos, porque es más probable que el número de vehículos sea menor. Para el resto de pedidos $i \in O$, $w_i = 1$.

Al igual que el problema en la sección 6.2, el problema anterior corresponde a un problema de flujo a coste mínimo. El grafo para este caso corresponde con el mismo grafo de esa sección con dos modificaciones:

1. Para cada arco cuyo pedido asociado esté en K_j, una unidad de flujo se extrae en el origen del arco y la misma unidad de flujo es insertada de nuevo en el nodo destino del arco. Esto es equivalente a forzar el flujo a través de ese arco, es decir, forzar la inclusión de ese pedido en la solución.

2. La segunda modificación impone la prioridad en la función objetivo de los pedidos pertenecientes a la solución original. Para cada arco cuyo pedido asociado esté en S_o el coeficiente de coste es sustituido por $-M_i$.

La figura 6.7 ilustra estos cambios. Para cada arco, el primer valor representa el coste por unidad de flujo y el segundo la capacidad del arco. Los pedidos i y s pertenecen a la solución original por lo que sus costes son $-M_i$ y $-M_s$. Los pedidos r y k no pertenecen a la solución original pero, en este caso, el pedido k es forzado en la solución ya que $K_j=\{k\}$. Esto se implementa extrayendo una unidad de flujo del nodo origen del arco correspondiente al pedido k, e inyectándolo de nuevo en el nodo destino de dicho arco.

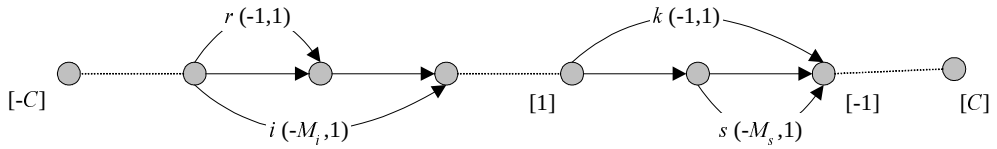


Figura 6.7: Grafo que chequea la admisibilidad del nodo $K=\{j\}$

6.3.2.2. Ilustración

En la figura 6.8 se muestra, para el ejemplo de la figura 6.1 con C = 1, la solución original para el número máximo de pedidos servidos. S_o= {A, C, D, F, G, H, J} y O={B, E, I}. Por tanto, $|S_o| = p^* = 7$. El número de vehículos requerido es $V(S_o)$= 3. La figura 6.9 muestra el árbol de exploración hasta el nivel 1.

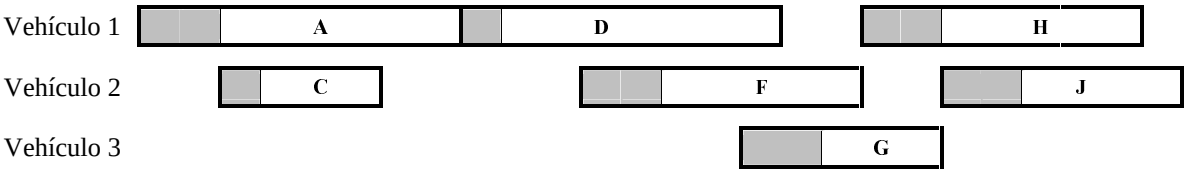


Figura 6.8: Solución original S_o con $p^*=7$ y $V(S_o)=3$

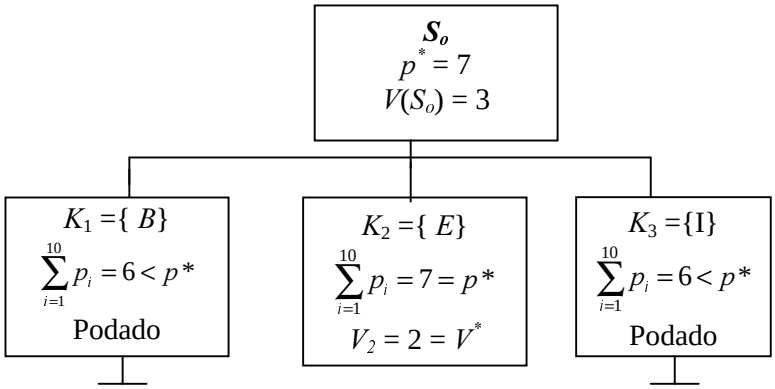


Figura 6.9: Árbol de exploración

En la figura 6.10 se muestra el grafo de flujo usado para chequear la admisibilidad del nodo de primer nivel K_1={B}, es decir, el nodo que representa forzar el pedido B para ser uno de los p^*=7 pedidos a servir. Nótese que en ese caso, sólo 5 pedidos más podrían ser servidos como máximo. Por tanto, no existe una solución óptima con p^* pedidos que contenga al conjunto K_1 y ese nodo puede ser podado.

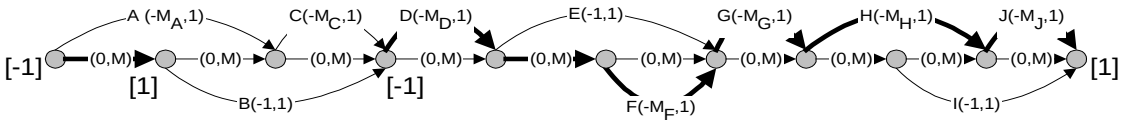


Figura 6.10: Grafo de flujo para chequeo de admisibilidad en el nodo K_1={B}

Sin embargo, una solución óptima sirviendo p^* = 7 pedidos y usando V_2=2 vehículos se encuentra en el nodo de primer nivel K_2={E}. Esa solución se muestra en la figura 6.11.

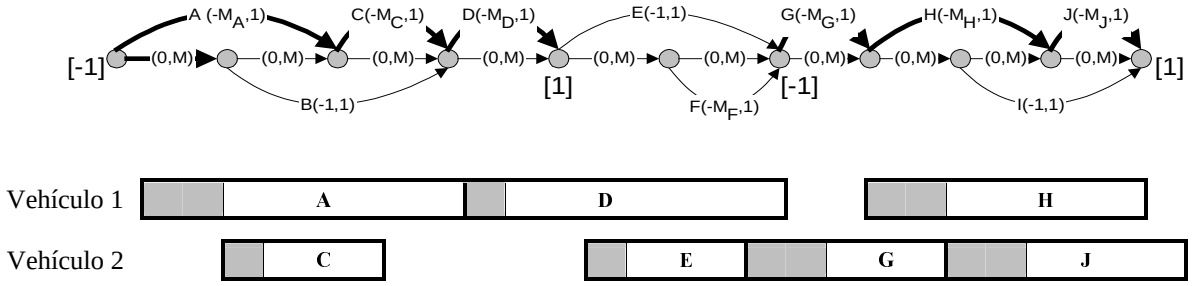


Figura 6.11: Solución óptima (p^* = 7 y V^*= V_2= 2)

Finalmente y puesto que el nodo K_3 = {I} de primer nivel ha sido podado, cualquier nodo de segundo nivel que contenga ese conjunto puede también ser podado. De esta forma, aunque el árbol tuviera en principio $2^{n-p^*}= 2^{10-7}= 8$ nodos, sólo tres nodos y un único nivel ha tenido que ser evaluado.

7.4. RESULTADOS COMPUTACIONALES

Se han considerado los siguiente parámetros para la generación aleatoria de problemas:

- Cuatro tamaños de problemas respecto del número de pedidos: 25, 50, 75 y 100 pedidos. Se generaron dos conjuntos o baterías de problemas dependientes del grado de simultaneidad de los pedidos. Estas baterías se denotan como PS_1 y PS_2. Los promedios de simultaneidad en la fase de producción (GSP) y en la fase de distribución (GSD) se muestran en la tabla 6.3.

Batería	*n*	GSP	GSD
PS_1	25	1.8	7.6
	50	2.0	9.2
	75	2.1	10.0
	100	2.0	9.5
PS_2	25	2.8	11.1
	50	2.6	12.8
	75	2.6	13.5
	100	2.6	13.8

Tabla 6.3: Grados de simultaneidad de las baterías

Como se muestra en la tabla 6.3, los grados de simultaneidad son mayores en la segunda batería de problemas.

- Los tiempos de la fase de producción se obtienen aleatoriamente dentro del rango [1,5]. Los tiempos de la fase de distribución respecto al intervalo [4,24].

- Diez instancias fueron generadas para cada tamaño de problema en ambas baterías.

- Todos los datos son promedios sobre las 10 instancias de cada tamaño de problema y todos los tiempos de ejecución vienen dados en segundos de CPU sobre un Intel Pentium III 850 MHz.

En una primera fase, todas las instancias de PS_1 fueron resueltas utilizando el método exacto con objeto de obtener la solución óptima y su tiempo de computación y de ese modo medir el comportamiento del procedimiento heurístico. Los valores considerados para la capacidad fueron 1, 2 y 3. La tabla 6.4 muestra los tiempos medios de proceso empleados por el método.

Las mismas instancias se resolvieron usando el enfoque heurístico propuesto, con cuatro valores diferentes para el número de niveles explorados en el árbol. *D* representa esos valores. La tabla 6.5 resume los resultados. En esa tabla se muestra, para cada valor de *D*, el número de problemas resueltos de forma óptima sobre las 10 instancias de cada tamaño y el tiempo medio de CPU. También se muestra el promedio de reducción en el número de vehículos de la solución óptima respecto a la solución inicial.

C	*n*	Tiempo medio	Valor medio p^*	Valor medio V^*
1	**25**	0.4	10.6	4.7
	50	4.6	22.5	6.3
	75	14.1	32.7	6.5
	100	15.7	44.6	6.4
2	**25**	12.6	16.9	7.0
	50	485.2	35.5	10.1
	75	907.2	53.0	10.6
	100	1444.6	72.9	10.3
3	**25**	57.0	20.9	10.5
	50	4871.1	43.1	12.6
	75	6312.2	65.7	13.0
	100	7412.5	89.8	12.8

Table 6.4. Resultados del método de resolución exacto sobre PS_1

		Solución inicial	*D* = 1		*D* = 2		*D* = 3		*D* = 4		
C	***n***	**Nº éxitos**	**Nº éxitos**	**Tiempo medio**	**Nº éxitos**	**Tiempo medio**	**Nº éxitos**	**Tiempo medio**	**Nº éxitos**	**Tiempo medio**	**Reducción media de vehículos**
1	**25**	4	9	0.1	9	0.2	**10**	0.2	**10**	0.6	0.7
	50	3	8	0.2	**10**	0.7	**10**	1.8	**10**	7.6	0.7
	75	4	8	0.4	9	2.6	**10**	13.2	**10**	81.9	0.6
	100	4	9	0.7	9	6.6	**10**	41.6	**10**	349.3	0.7
2	**25**	8	**10**	1.1	**10**	3.2	**10**	6.4	**10**	10.1	0.2
	50	7	8	1.5	9	7.3	9	26.2	**10**	73.3	0.4
	75	6	**10**	2.7	**10**	18.5	**10**	95.2	**10**	385.6	0.4
	100	6	**10**	3.9	**10**	30.5	**10**	164.7	**10**	865.5	0.4
3	**25**	7	**10**	0.5	**10**	1.2	**10**	1.9	**10**	2.2	0.3
	50	8	**10**	1.0	**10**	2.7	**10**	5.9	**10**	10	0.2
	75	6	**10**	1.3	**10**	5.8	**10**	14.6	**10**	33.4	0.4
	100	8	9	1.7	**10**	8.8	**10**	23.3	**10**	65.4	0.3

Tabla 6.5: Resultados de la heurística sobre PS_1 (negrita corresponde con un 100% éxito)

Como se muestra en la tabla 6.5, la solución óptima se alcanza en la mayoría de los problemas. De los 120 problemas considerados en el experimento, la solución óptima se encontró en 111, 116 y 119 casos para uno, dos y tres niveles de exploración respectivamente, alcanzando el 100% de éxito para $D = 4$. El número de vehículos que se reducen nunca es mayor de 2. La solución inicial proporciona la solución óptima en 71 de las 120 ejecuciones.

Únicamente cuando la capacidad C fue igual a 1 y el número de niveles explorados mayor de 2, los tiempos de proceso del método de solución exacta fueron un poco mejor que los tiempos de la heurística. Sin embargo, con C=2 y C=3 el método heurístico alcanzó una reducción apreciable en los tiempos de proceso.

Los resultados también muestran una mejora del procedimiento heurístico, tanto en tiempo como resultados, conforme aumenta la capacidad de producción C. Por ello, se ejecutaron los problemas de la segunda batería con $C = 1$ y de ese modo poder evaluar el comportamiento del método con un aumento del solapamiento de los pedidos. Esta segunda batería produce un remarcable aumento de los tiempos de computación del método de resolución exacto. Los resultados se presentan en la tabla 6.6.

Los resultados obtenidos para PS_2 son también satisfactorios. Todas las soluciones óptimas excepto una fueron obtenidas con D = 3. Aunque no se muestra en la tabla, el 100% de éxitos se obtuvo de nuevo para D = 4. Al igual que con la batería PS_1, el número de vehículos reducido desde la solución inicial estuvo dentro del intervalo [0,2]. Los tiempos de computación fueron también similares a los obtenidos para PS_1.

		Método exacto			Solución inicial	Heurística					
						$D = 1$		$D = 2$		$D = 3$	
C	n	Tiempo medio	p^* medio	V^* medio	Nº éxitos	Nº éxitos	Tiempo medio	Nº éxitos	Tiempo medio	Nº éxitos	Tiempo medio
1	25	27.6	8.4	6.4	8	10	0.1	10	0.4	10	0.7
	50	6648.3	19.1	8.8	3	8	0.2	9	0.9	9	5.1
	75	13618.3	29.8	8.8	2	7	0.4	9	2.7	10	22.2
	100	35964.8	41.8	10.0	4	8	0.7	10	6.0	10	49.3

Tabla 6.6. Resultados de la batería PS_2

Capítulo 7

Escenario II: El problema PDP con ventanas temporales de entrega y una planta de producción

Índice

7.1. INTRODUCCIÓN

En el Escenario II se acomete el estudio del problema de la planificación conjunta de la producción y distribución de pedidos con ventanas temporales de entrega y una planta de producción. El problema vuelve a ser el de seleccionar y planificar una serie de pedidos para ser procesados en una planta de fabricación e inmediatamente distribuidos a la localización del cliente.

El capítulo está estructurado del siguiente modo:

- En el apartado 7.2 del capítulo se describe y formula el problema.

- En el apartado 7.3 se describe brevemente el método utilizado para la obtención de soluciones exactas.

- En los apartados 7.4 y 7.5 se presentan los métodos de resolución heurísticos implementados.

- Finalmente, en el apartado 7.6 se presentan los resultados de los experimentos.

7.2. DESCRIPCIÓN DEL PROBLEMA

La planificación coordinada de la producción y distribución de pedidos con ventanas temporales se ocupa del problema de seleccionar y planificar una serie de pedidos para ser procesados en una planta de fabricación e inmediatamente distribuidos a la localización del cliente, teniendo en cuenta que cada pedido posee una ventana temporal dentro de la que debe efectuarse la entrega. No se admiten entregas fuera de ese intervalo o ventana temporal.

El estudio parte de una situación totalmente determinista en la que se conoce el calendario de pedidos a abastecer durante un horizonte temporal determinado.

Fase de Producción del pedido

Para la fabricación de los pedidos se dispone de una única planta con capacidad de producción limitada *C*. Consideramos capacidad de producción como el número de pedidos que pueden ser preparados simultáneamente, es decir, la fabricación de un

pedido se considera un proceso continuo que requiere una unidad de capacidad durante el tiempo que dure el proceso.

Fase de Distribución del pedido

En la etapa de distribución de un pedido se consideran tres fases consecutivas: el envío del pedido hasta el lugar correspondiente, la descarga del mismo y la vuelta del vehículo a la planta. Cada vehículo puede transportar cualquier pedido, pero no más de un pedido en un mismo viaje. También se asume que el tamaño del pedido es menor que la capacidad del vehículo. Por todo ello, la fase de distribución de un pedido puede considerarse como un único proceso, que se realiza sin interrupción, y que comienza inmediatamente después del fin de la fase de producción del pedido.

Además, como todos los tiempos de proceso (producción y distribución) son conocidos, la ventana temporal de entrega puede ser trasladada a una ventana temporal de comienzo de la actividad asociada con el pedido. Un instante ideal de entrega se asume dentro de la ventana temporal. El esquema de la actividad de un pedido en este escenario se resume en la Figura 7.1.

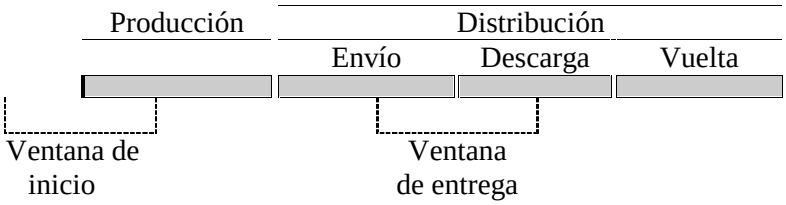


Figura 7.1: Actividad de un pedido

Al igual que en el Escenario IA, en este escenario se plantea una situación en la que los recursos son limitados tanto en la fase de producción como de distribución. Por ello y debido a que en el problema se impone que los pedidos deben servirse dentro de su ventana temporal, se considerará como objetivo en el problema la maximización del valor de los pedidos servidos, asumiendo que no todos los pedidos tienen por qué que ser atendidos y que la entrega de un pedido en un instante distinto al ideal supone un decremento del valor del pedido directamente proporcional a la desviación sobre dicho instante, considerando para ello índices correctores sobre el valor del pedido, para el caso de adelanto y retraso sobre el instante ideal de entrega.

Como ya se ha comentado en capítulos anteriores, en términos de la teoría de secuenciación de trabajos, la planificación coordinada de producción y distribución de pedidos con ventanas temporales corresponde a un problema de taller de flujo con dos estaciones formadas por máquinas paralelas, asumiendo *no espera* entre procesos (FSPM con *no-wait*). La primera estación correspondería a la planta de

fabricación, la cuál está compuesta de un número de máquinas idénticas, igual a la capacidad de la planta. La segunda estación está compuesta por un número de máquinas idénticas igual al número de vehículos. Cada trabajo correspondería con la producción y distribución de cada pedido. El objetivo está íntimamente relacionado con la maximización de los pesos de los trabajos servidos en su fecha de entrega, si bien ahora, podemos adelantar o retrasar el pedido sobre esta fecha, aunque siempre dentro de la ventana temporal. Un problema también relacionado con éste sería el problema de la planificación de trabajos variables (VSP). Sin embargo, estos problemas consideran una única estación de máquinas para el procesamiento de los trabajos, por lo que se ajustarían al caso particular de disponer de capacidad ilimitada de producción, o bien de un número ilimitado de vehículos.

- **Notación**

Los datos básicos de cada pedido i, $i = 1 \ldots n$, son los siguientes:

- e_i : instante ideal de entrega.
- $[A_i, B_i]$: Ventana temporal dentro de la que debe entregarse el pedido.
- W_i: valor o beneficio obtenido con la entrega del pedido.
- w_i^- : penalización por unidad de tiempo de entrega adelantada respecto al instante ideal.
- w_i^+: penalización por unidad de entrega retrasada respecto al instante ideal.

En la figura 7.2 se completan los datos asociados a cada pedido en el problema.

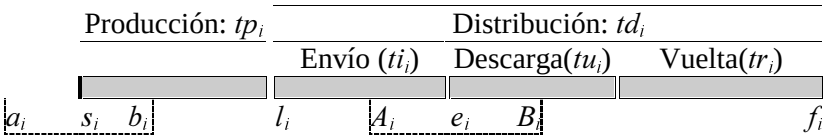


Figura 7.2:. Datos asociados a un pedido

- **Ilustración**

La tabla 7.1 muestra los datos de 5 pedidos. La solución óptima para $C = 1$ y $V = 2$ se representa en la figura 7.3.

	tp_i	td_i	s_i	a_i	b_i	W_i	w_i^-	w_i^+
Pedido	2	6	3	2	3	12	1	1
Pedido	2	11	4	3	4	20	1	1
Pedido	2	6	4	3	5	10	1	1
Pedido	3	6	13	12	14	13	2	1
Pedido	1	5	16	15	17	10	1	1

Tabla 7.1: Ejemplo para Escenario II

La solución óptima del problema procesa los pedidos 1,3, 4 y 5. El pedido 3 se retrasa un periodo sobre su instante ideal de entrega. El beneficio obtenido fue 44.

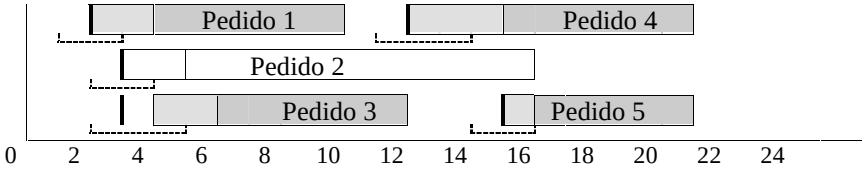


Figura 7.3: Solución óptima del ejemplo de Tabla 7.1

7.3. FORMULACIÓN DEL PROBLEMA

Desde una perspectiva asociada a problemas de planificación de trabajos, el problema puede ser definido como sigue: Considérese el problema de encontrar una planificación óptima para un conjunto de n trabajos (pedidos), denotado por $N = \{1,2,...,n\}$, sobre un taller de flujo flexible con no espera en proceso. El taller consiste de dos centros o estaciones de máquinas (fase de producción y fase de distribución) donde el centro de producción está equipado con $C \geq 1$ idénticos recursos o máquinas, que corresponden con la capacidad de la planta. Por otra parte, el centro 2 o centro de distribución posee V máquinas idénticas que se corresponden con la flota de vehículos. Se asocia con cada trabajo una ventana temporal $[a_i,b_i]$ para el comienzo de la actividad asociada con el trabajo, dentro de la cuál se incluye un instante ideal de comienzo s_i. Cada trabajo i tiene un valor positivo W_i reflejando el beneficio obtenido por realizar el trabajo. Este valor W_i asume que el comienzo del trabajo se produce en su instante s_i . Si el instante de comienzo del trabajo fuera anterior (o posterior) a s_i un índice corrector w_i^- (o w_i^+) se aplicaría sobre la desviación para penalizar el beneficio obtenido con el mismo.

Los trabajos seleccionados deben ser procesados de forma continua desde su inicio en el centro de producción hasta su finalización en el centro de distribución, sin ninguna interrupción en los centros, ni entre ellos. De ese modo, el trabajo $i \in N$, consiste de una secuencia de 2 operaciones, cada una de ellas correspondiente al procesamiento del trabajo i durante un tiempo de proceso tp_i en el centro de

producción y td_i en el centro de distribución.

Para modelar el problema consideramos la formulación descrita en [Gerstback, 1976] para la planificación de trabajos variables. Introducimos una escala de tiempo discreto, dividiendo el tiempo en T periodos o instantes, con $j = 1...T$. Considerando $[a_i,b_i]$ la venta temporal de comienzo del pedido i, es claro que a_i y $d_i = b_i + tp_i$ representan el instante más temprano de comienzo y el último instante posible de finalización de la fase de producción del pedido i, respectivamente. Por tanto, el conjunto de intervalos en los que el pedido i podría ser procesado en el centro de producción es $P_i = \{a_i,\ldots,d_i\text{-}1\}$. Por otro lado, dados $h_i = a_i + tp_i$ y $v_i = b_i + tp_i + td_i$, $D_i=\{h_i,\ldots,v_i\text{-}1\}$ representa el conjunto de instantes sobre el horizonte de planificación durante los cuales el pedido i podría ser procesado en el centro de distribución. Además de esto, para cada pedido i, sea I_i el conjunto de instantes correspondientes a la intersección de P_i y D_i, $I_i = \{h_i,..,d_i\text{-}1\}$. La figura 7.4 muestra estos datos.

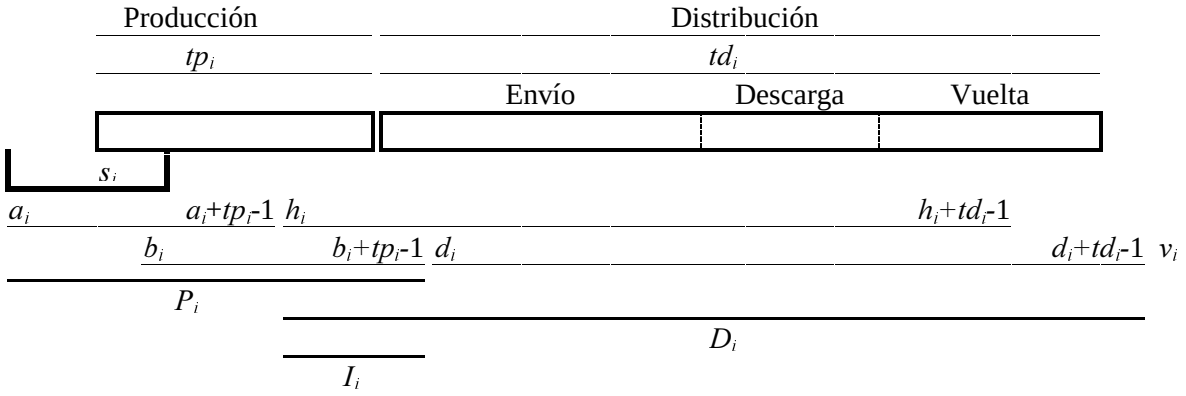


Figura 7.4: Datos del pedido i asociados al modelo

Sea J_j^P el conjunto de trabajos cuya fase de producción podría producirse en el periodo j. De la misma forma, J_j^D define el conjunto de trabajos con fase de distribución asignable al periodo j. Estos conjuntos son usados para asegurar que en cualquier instante, el número total de trabajos asignados en los centros de producción y distribución no excedan del número de recursos disponibles.

Nuestro modelo incorpora varios tipos de variables de decisión relacionadas con el tiempo de proceso de cada trabajo:

y_{ij} = 1 si el trabajo i comienza en el instante j ; 0 en otro caso. $j \in P_i$

u_{ij} = 1 si el trabajo i es procesado en el centro de producción en el instante j ; 0 en otro caso. $j \in P_i$

v_{ij} = 1 si el trabajo i es procesado en el centro de distribución en el instante j ; 0 en otro caso. $j \in D_i$

La suma de las variables u_{ij} con $j \in P_i$ debe ser igual al tiempo de proceso en planta tp_i si el pedido es servido. De la misma forma, la suma de variables v_{ij} con $j \in D_i$ debe ser igual al tiempo de distribución td_i. Para determinar los trabajos completados definimos las siguientes variables de decisión:

x_i = 1 si el trabajo i es procesado; 0 en otro caso.

Obviamente, si x_i = 0 las variables x_{ij} u_{ij} y v_{ij} tomarán valor 0 para cualquier periodo j.

Finalmente, definimos las variables n^+_i y n^-_i que recogen la desviación sobre el instante ideal de comienzo de los trabajos seleccionados. Si el trabajo i comienza sobre su instante ideal, ambas variables tomarán valor nulo.

- **Restricciones del modelo**

1) En el centro de producción, a cada trabajo se le puede asignar un número de periodos igual a su tiempo de producción.

$$\sum_{j \in P_i} u_{ij} = \quad tp_i x_i \qquad i = 1 \ldots n$$

2) En el centro de distribución, a cada trabajo se le puede asignar un número de periodos igual a su tiempo de distribución.

$$\sum_{j \in D_i} v_{ij} = \quad td_i x_i \qquad i = 1 \ldots n$$

3) Los periodos asignados en la fase de producción tienen que ser adyacentes.

$$tp_i u_{ij} - tp_i u_{ij+1} + \sum_{k=j+2}^{d_i - 1} u_{ik} \leq tp_i \qquad i = 1 \ldots n; \ \ \forall j \in P_i$$

4) Los periodos asignados en la fase de distribución tienen que ser adyacentes.

$$td_i v_{ij} - td_i v_{ij+1} + \sum_{k=j+2}^{v_i - 1} v_{ik} \leq td_i \qquad i = 1 \ldots n; \;\; \forall j \in D_i$$

5) Estas restricciones imponen el instante de comienzo de un trabajo.

$$(u_{ij} + v_{ij}) - u_{ij-1} - v_{ij-1} = y_{ij} \qquad i = 1 \ldots n; \;\; \forall j \in P_i \qquad j \neq a_i$$

6) Aseguran que no más de C trabajos en la fase de producción son asignados a cualquier periodo.

$$\sum_{i \in J_j^P} u_{ij} \leq C \qquad j = 1 \ldots T$$

7) Permiten a lo sumo V trabajos procesándose en la fase de distribución durante cualquier instante de tiempo.

$$\sum_{i \in J_j^D} v_{ij} \leq V \qquad j = 1 \ldots T$$

8) y 9): Obligan a que la etapa de distribución de cada trabajo comience, sin retraso, justo después del último periodo en el que finaliza la fase de producción del trabajo.

$$u_{ij} + v_{ij} \leq 1 \qquad i = 1 \ldots n; \forall j \in I_i$$

$$u_{ij} - (u_{ij+1} + v_{ij+1}) \leq 0 \qquad i = 1 \ldots n; \forall j \in I_i$$

10) Se define, para los trabajos que se procesan, el número de periodos en los que se adelanta o retrasa el trabajo sobre su instante ideal de comienzo s_i.

$$(\sum_{j \in P_i} j y_{ij} - s_i x_i) + n_i^- - n_i^+ = 0 \qquad i = 1 \ldots n$$

11) Se definen los periodos de tiempo en los que cada trabajo podría ser procesado.

$$u_{ij} \;\; \forall j \in P_i; v_{ij} \;\; \forall j \in D_i \qquad i = 1 \ldots n$$

El modelo completo que maximiza el valor total de los trabajos procesados sería el siguiente:

$$Max \quad \sum_{i=1}^{n}\left(w_i x_i - w_i^- n_i^- - w_i^+ n_i^+\right)$$

s.a.

$$\sum_{j\in P_i} u_{ij} = \quad tp_i x_i \qquad i=1\ldots n \tag{1}$$

$$\sum_{j\in D_i} v_{ij} = \quad td_i x_i \qquad i=1\ldots n \tag{2}$$

$$tp_i u_{ij} - tp_i u_{ij+1} + \sum_{k=j+2}^{d_i-1} u_{ik} \le tp_i \qquad i=1\ldots n;\ \forall j \in P_i \tag{3}$$

$$td_i v_{ij} - td_i v_{ij+1} + \sum_{k=j+2}^{v_i-1} v_{ik} \le td_i \qquad i=1\ldots n;\ \forall j \in D_i \tag{4}$$

$$(u_{ij} + v_{ij}) - u_{ij-1} - v_{ij-1} = y_{ij} \qquad i=1\ldots n;\ \forall j \in P_i \quad j \ne a_i \tag{5}$$

$$\sum_{i\in J_j^P} u_{ij} \le C \qquad j=1\ldots T \tag{6}$$

$$\sum_{i\in J_j^D} v_{ij} \le V \qquad j=1\ldots T \tag{7}$$

$$u_{ij} + v_{ij} \le 1 \qquad i=1\ldots n; \forall j \in I_i \tag{8}$$

$$u_{ij} - (u_{ij+1} + v_{ij+1}) \le 0 \qquad i=1\ldots n; \forall j \in I_i \tag{9}$$

$$\left(\sum_{j\in P_i} j y_{ij} - s_i x_i\right) + n_i^- - n_i^+ = 0 \qquad i=1\ldots n \tag{10}$$

$$u_{ij} \quad \forall j \in P_i;\ v_{ij} \quad \forall j \in D_i \qquad i=1\ldots n \tag{11}$$

$$x_i, y_{ij}, u_{ij}, v_{ij} \in \{0,1\} \quad n_i^+, n_i^- \ge 0$$

7.4. MÉTODO EXACTO DE RESOLUCIÓN

Para medir el comportamiento de las heurísticas descritas en la siguiente sección, inicialmente se ha implementado un procedimiento exacto para obtener las soluciones óptimas de los problemas. El método de resolución exacto está basado en el método exacto propuesto para el problema PDP en el Escenario IA. El método construye un grafo que colecciona todas los estados admisibles que pueden aparecer en la programación de los pedidos, a través de un simple método de evaluación de la capacidad de producción y el número de vehículos necesarios en cada estado. El valor del camino máximo desde el nodo de salida al nodo final del grafo es la

solución óptima del problema. El proceso se vuelve ineficiente en tiempo y capacidad de memoria cuando el número de pedidos aumenta por encima de 50.

A partir del método diseñado para el problema PDP en el Escenario IA, las modificaciones que se incluyen en la construcción del grafo son las siguientes:

El conjunto de estados N_i se divide en subconjuntos $N^r{}_i$ que recogen los estados generados para cada pedido *i* en su instante posible de comienzo *r*. Por tanto, para cada ventana temporal $[a_i,b_i]$ el número de subconjuntos $N^r{}_i$ asociados a cada pedido *i*se corresponde con la amplitud de la ventana $(b_i - a_i + 1)$.

A la hora de explorar el grafo para determinar todos los estados de un subconjunto N_i^r, se descartan subconjuntos de estados correspondientes al mismo pedido *i*. Para la construcción progresiva del grafo, inicialmente se ordenan todos los instantes de comienzo de todos los pedidos. En función de esa ordenación se van creando los conjuntos de estados N_i^rutilizando la misma filosofía que en el Escenario IA del problema. Aquí también es posible usar la regla de reducción de estados, durante el proceso de construcción.

El camino máximo entre el estado inicial y el estado final proporciona la planificación óptima de los pedidos. El pseudocódigo del método de construcción sería el siguiente:

Procedimiento Construcción_Grafo()

N_i^r = Conjunto de estados creados al comienzo del procesamiento del pedido *i* en instante el *r*

Para cada pedido *i*

Para cada instante posible de comienzo para el pedido *i,* $r \in [a_i,b_i]$

Para cada estado previo N_j^d , $j \neq i$

Evaluación de admisibilidad con objeto de crear un estado con los pedidos de N_j^d que continúan procesándose en el instante *r*, más el pedido *i*.

Unir ambos estados con una arco cuyo peso es el valor del pedido *i* en el instante de comienzo *r*.

Fin Para

Fin Para

Fin Para

Fin

Veamos un pequeño ejemplo del método:

Pedidos	a_i	s_i	b_i	tp_i	td_i	W_i	w^+_i	w^-_i
A	2	3	4	2	16	5	1	1
B	3	4	4	2	4	6	1	1
C	5	5	5	1	11	8	1	1
D	15	15	16	1	7	5	1	1

Tabla 7.2: Datos de ejemplo para el método exacto

El esquema temporal de los pedidos es el siguiente:

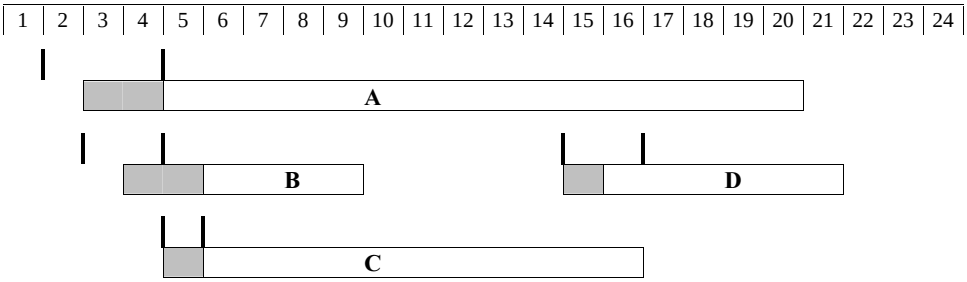


Figura 7.5: Representación temporal de los pedidos de la tabla 7.2

El grafo que resulta tras la aplicación del algoritmo es el siguiente:

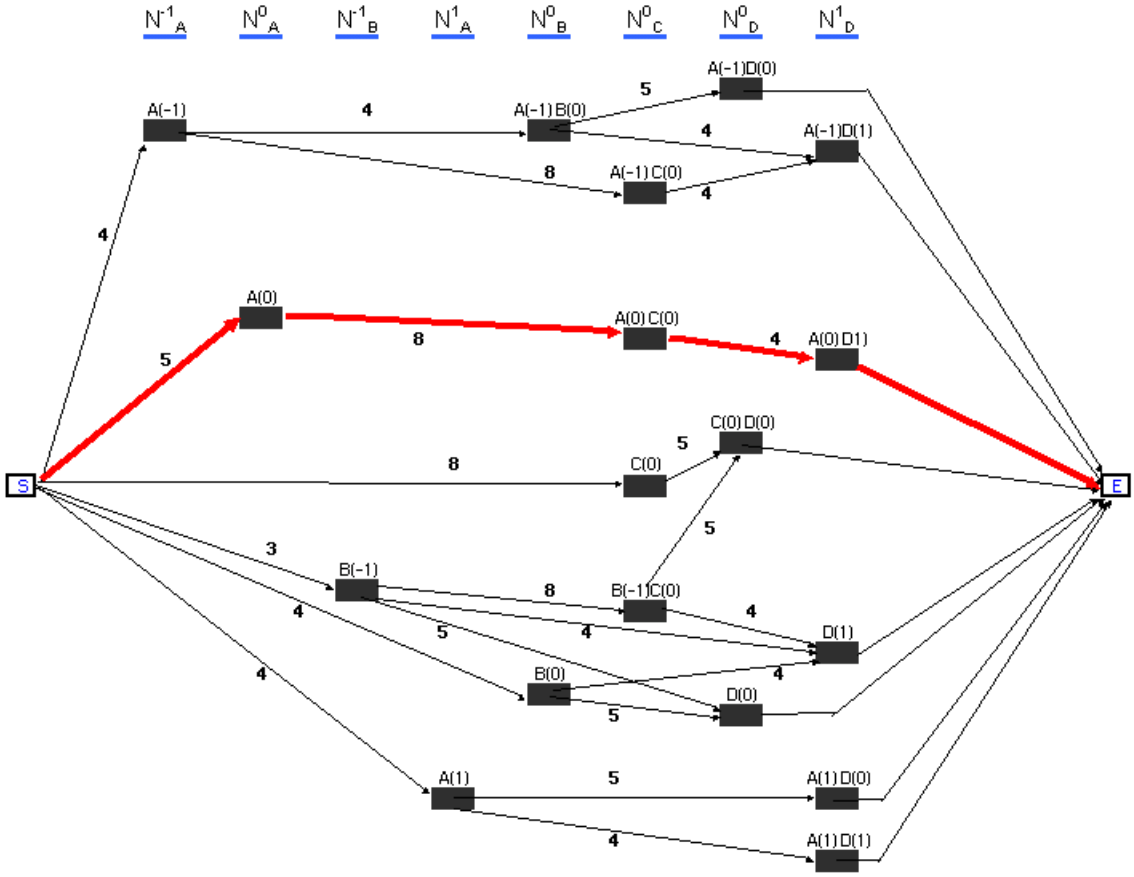


Figura 7.6: Grafo correspondiente al ejemplo de la tabla 7.2

La solución óptima procesa los pedidos A, C y D, éste último con un periodo de retraso. El valor de la función objetivo es 17.

- **Complejidad**

Al igual que en el método de resolución exacto del Escenario IA, la complejidad se obtiene como la complejidad del proceso de construcción de un estado multiplicado por el número de estados. La complejidad asociada a un estado sigue siendo la misma, es decir, $O((C + V) \log (C + V))$.

Sin embargo, el número de estados crece notablemente, pues hay que contabilizar todos los instantes de comienzo de cada pedido.

Supongamos una amplitud k_i para el pedido i $k_i = b_i - a_i + 1$. Eso significa k_i posibles instantes de comienzo para el pedido i. Suponiendo la limitación de recursos en ambas fases de la actividad del pedido que impide el procesamiento de todos ellos, el cálculo del orden en el número de nodos y arcos del grafo sería análogo al realizado para el grafo definido en el Escenario IA del problema. Si en ese escenario existían n posibilidades de combinación, en este escenario serían del orden de nk, k = máximo$\{k_i\}$. Por tanto el orden en el número de nodos y arcos sería $O\left((nk)^{C+V}\right)$

La complejidad total del proceso de construcción es por tanto de:

$$O((C + V) \log (C + V)\ \left(nk\right)^{C+V}\ )$$

7.5. ALGORITMO DE BÚSQUEDA TABÚ

En esta sección se describe el primero de los métodos heurísticos propuestos para resolver el problema. El algoritmo está basado en la búsqueda tabú propuesta para el problema PDP en el Escenario IA. Los parámetros del algoritmo son descritos a continuación:

- **Solución Inicial:** Para obtener una solución inicial, consideramos un algoritmo GRASP basado en el método GRASP descrito en el Escenario IA. En este caso, en cada iteración del algoritmo asumimos que el instante de inicio de un pedido, que no pertenece a la solución que está siendo construida, es el instante dentro de su ventana temporal que proporciona el mejor valor admisible.

- **Movimientos y estructura de la vecindad:** Dada una solución s sea N(s) el

conjunto de todas las soluciones admisibles que pueden ser obtenidas desde *s* usando uno de los siguientes movimientos:

- *Cambio*: Una solución vecina de *s* se obtiene por un procedimiento de cambio donde un pedido *i* perteneciente al conjunto solución es reemplazado por otro u otros pedidos que no están en el conjunto solución. Denotamos por *S* el conjunto que contiene los pedidos que pertenecen al conjunto solución actual. Dado un pedido $i \in S$, consideramos todos los pedidos *j*, $j \notin S$, cuya fase de producción o de distribución coincide en algún instante con la del pedido *i*. Usemos H_i para denotar el conjunto de esos pedidos. Los pedidos pertenecientes a H_i son primero ordenados por valor y tentativamente introducidos en el conjunto solución *S* para chequear si las restricciones del problema siguen cumpliéndose. En el proceso de evaluación se examina cualquier posible instante de comienzo del pedido en orden al valor asociado a cada uno. La figura 7.7 ilustra este movimiento para el caso en que reemplazamos de la solución el pedido B.

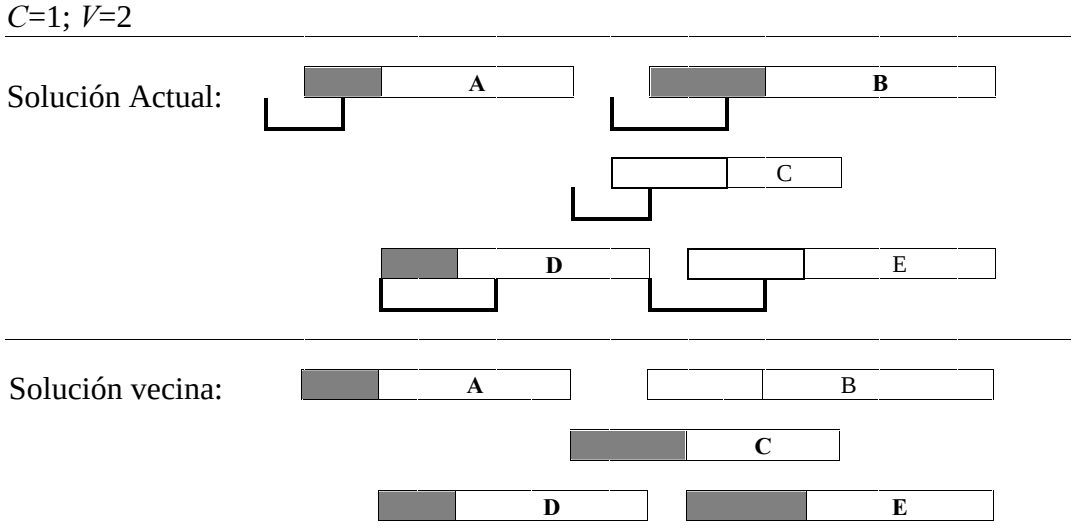


Figura 7.7: Movimiento de cambio para el pedido B

Los pedidos que entran en la solución pasan a ser tabú durante un número determinado de iteraciones.

- *Quitar*: Para permitir la extracción de pedidos de la solución sin introducir ninguno, cada cierto número de iteraciones se considera el movimiento de quitar un pedido de la solución. Estos pedidos pasan a formar parte de una segunda lista tabú que se actualiza cada vez que se realiza este tipo de movimiento. Este tipo de movimiento lleva, lógicamente, a soluciones peores. Sin embargo, llega a ser útil para escapar de máximos locales y diversificar la búsqueda.

- **Lista tabú:** El tamaño de la lista tabú es un parámetro muy importante en un algoritmo de búsqueda tabú. La lista tabú puede ser fija o variable. [Glover et al, 1993] sugieren el uso de listas tabú variables para obtener mejores resultados. Nosotros intentamos una estrategia donde el tamaño de la lista tabú varía para ajustar el número de iteraciones en las que se acepta un movimiento de quitar un pedido. El movimiento de quitar un pedido sólo ocurría cuando todos los pedidos pertenecientes al conjunto solución son tabú. Por tanto, el número de pedidos en la solución determinará la frecuencia de movimientos de tipo quitar aceptados para un tamaño fijo de la lista tabú. Los experimentos preliminares mostraron que los mejores resultados se obtenían para un tamaño de lista igual al número de pedidos en la solución inicial dividido por dos. El estudio realizado con esta implementación concluyó que era conveniente que este movimiento no se realizase con una frecuencia menor al doble del número de pedidos del problema, por lo que se estableció este valor como la frecuencia exacta para realizar un movimiento de extracción.

- **Criterio de aspiración:** Para el criterio de aspiración usamos la forma más clásica en la que un movimiento tabú es aceptado si produce una solución mejor que la mejor solución obtenida hasta ese momento.

- **Criterio de parada:** El algoritmo se detiene después de un número máximo de iteraciones. En los experimentos se establecieron 5000 iteraciones.

7.6. RESOLUCIÓN MEDIANTE ALGORITMOS GENÉTICOS

En este apartado se describen 2 estrategias basadas en algoritmos genéticos para la resolución del problema. La primera de las estrategias se basa en una implementación estándar de un algoritmo genético. La segunda implementación incide en las características propias del problema a la hora de establecer los elementos del algoritmo.

7.6.1. Algoritmo genético I

La descripción del algoritmo genético pasa por detallar los elementos del algoritmo y el funcionamiento general del mismo. Algunos de los elementos serán estudiados con distintos valores con objeto de determinar la configuración óptima del método. Todo este análisis se realiza posteriormente en la sección dedicada a la experimentación.

- **Representación individual: Genotipo y Fenotipo**

Un importante concepto es la separación estricta de las restricciones del problema y el método de evolución. Tal separación resulta en dos vistas completamente diferentes de un individuo. Por un lado, existe la representación de fenotipo para el problema con objeto de evaluar los individuos según su función fitness. Por otro lado, la representación de genotipo es aquella sobre la que el algoritmo genético trabaja, es decir, la representación codificada sobre la que se aplican los operadores de evolución.

Definición del genotipo

Para representar el genotipo de un individuo usaremos un *string* o cadena. La codificación usada para representar el individuo es una codificación clásica basada en permutaciones. La longitud del *string* corresponde con el número de pedidos. El valor de cada posición del vector es un número de pedido. Cada vector representa un orden de prioridad de proceso de los pedidos.

Estructura general de un individuo:

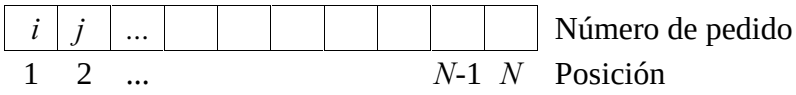


Figura 7.8 Estructura general de un individuo

Definición del fenotipo

El fenotipo se representa por un doble vector con una casilla para cada pedido. El pedido corresponde con el número de la casilla. Los valores del vector son los instantes de comienzo de los pedidos que son procesados así como el valor del pedido para ese instante. Un valor 0 en una posición indica que el pedido correspondiente a esa casilla no se realiza.

La decodificación del genotipo sigue un proceso secuencial en el que se chequea cada uno de los pedidos según el orden en el string. Cada pedido es tentativamente introducido en la solución a partir de su instante ideal. Si se violan las restricciones de capacidad o número de vehículos, entonces se continúan chequeando cada uno de los instantes de su ventana temporal, en orden decreciente al valor que proporciona el pedido en cada uno de esos instantes.

Fenotipo de un individuo:

i	*j*	*k*							
14	0	18						20	0
5	-	36		...				21	

Valor del pedido

Instante de comienzo

Figura 7.9: Estructura general del fenotipo

En este ejemplo el valor de pedido *i* que encontramos en el genotipo es *14* y su instante de comienzo es el *5.* El pedido *j* no se realiza, por tanto su valor es cero y no tiene instante de comienzo. El valor del pedido *k* es *18* y su instante de comienzo el *36*, y así sucesivamente.

- ***Fitness***

El *fitness* es la valoración de la solución del problema que representa cada individuo, es decir, es el valor de la función objetivo para esa solución. Nuestro *fitness* se computa como la suma de los valores, según instante de comienzo, de los pedidos realizados. Por tanto los valores de estos pedidos son:

- Cero si el pedido no se realiza, ya que su aportación a la función objetivo es nula.

- W_i si el pedido se realiza y además es entregado en el instante ideal

- $W_i + (r - s_i)w_i^-$ si el pedido se inicia en el instante $r \in [a_i , b_i]$ con $r < s_i$

- $W_i + (s_i - r)w_i^+$ si el pedido se inicia en el instante $r \in [a_i , b_i]$ con $r \geq s_i$

- **Población inicial**

La población inicial está compuesta por un conjunto de *m* individuos iniciales. Para la generación de los individuos iniciales se siguen tres estrategias:

- Generación aleatoria de permutaciones: Este método no considera el valor de los pedidos a la hora de establecer las prioridades.

- Generación basada en una elección avariciosa: Para conseguir unos individuos iniciales de partida sin tener en cuenta los valores de los pedidos a la hora de establecer prioridades se crean los individuos con la siguiente regla: Se ordenan los pedidos según W_i y en un proceso iterativo de *n* pasos, se escoge siempre aleatoriamente un pedido de entre los 5 mejores, según la ordenación establecida, y se incorpora al vector de prioridades. De esta manera, en los

individuos de partida los pedidos con mayor W_i tienen una mayor probabilidad de aparecer antes, y como el orden de los pedidos en el individuo es entendido como un orden de prioridad a la hora de ser realizados, tendrán una mayor probabilidad de ser servidos.

- Implementación híbrida: Parte de los individuos son creados aleatoriamente, y el resto con la estrategia de elección avariciosa.

El tamaño de la población es también otro parámetro de estudio de nuestro algoritmo, cuya dimensión se determinará en función de los resultados obtenidos. El algoritmo será probado con distintos tamaños y será elegido aquel que proporcione las mejores soluciones.

- **Operadores**

Mutación

Construye un nuevo individuo a partir de otro existente en la población a través de un procedimiento *swap*. Se elige aleatoriamente un individuo *j* entre los existentes. Se toman dos posiciones cualesquiera del individuo y se intercambia la información contenida en ellas, es decir, se intercambia la prioridad de dos pedidos.

Ind. Inicial	3	2	6	7	10	8	5	9	1	4

Ind. Final	3	2	6	5	10	8	7	9	1	4

Figuras 7.10. Ejemplo del operador mutación

Cruce

Crea un nuevo individuo (hijo) a partir de dos individuos de la población (padres). La elección de estos individuos, al igual que en el caso de la mutación, se realiza al azar. El cruce elegido es el COX [Goldberg, 1989]. El funcionamiento de este cruce es el siguiente: la cadena del padre 2 se copia en el individuo hijo. Se toma una subcadena central en el padre 1. Se buscan en la cadena hijo los valores existentes en la subcadena central del padre 1. Las posiciones donde aparecen estos valores se dejan en blanco. Tenemos por tanto la cadena hijo con una serie de huecos. A continuación se desplazan los valores en la cadena hijo hacia los extremos, de forma que los huecos queden situados en las posiciones centrales del individuo. Finalmente, los valores de la subcadena central del padre 1 se copian en la subcadena central, en los huecos existentes en el hijo. La siguiente figura ilustra el procedimiento.

Padre 1	3	2	6	7	10	8	5	9	1	4
Padre 2	5	7	4	10	2	1	3	9	6	8
Hijo	5	7	4	10	2	1	3	9	6	8
Hijo	5		4		2	1	3	9	6	
Hijo	5	4	2				1	3	9	6
Hijo	5	4	2	7	10	8	1	3	9	6

Figura 7.11: Ejemplo del operador COX

En cada iteración del algoritmo se escogerá, en función de una probabilidad, el operador a aplicar a la población. El valor de las probabilidades de cada operador es otro de los parámetros de estudio del algoritmo.

- **Tipo de reproducción**

La técnica de reproducción de la población elegida es la reproducción *steady-state*, reemplazando un individuo en cada iteración. El elemento que se elimina de la población es el de peor *fitness*.

- **Criterio de parada**

El criterio de parada vendrá dado por un número de iteraciones. Éste es otro parámetro del algoritmo, será un número lo suficientemente alto como para obtener buenas soluciones, pero tal que su aumento no proporcione una mejora muy apreciable en las soluciones obtenidas.

- **Funcionamiento del algoritmo**

Una vez definido el tamaño de la población, la probabilidad de cruce, la probabilidad de mutación y el número de iteraciones, el funcionamiento sería el siguiente:

Se crea un número de individuos iniciales igual al tamaño de la población, y se halla el *fitness* de cada uno de ellos con el fin de valorar la bondad de la solución.

El siguiente paso es la creación de un nuevo individuo, según la elección entre cruce y mutación. Una vez calculado el nuevo individuo se calcula su *fitness*. Se pasa a comparar el valor del *fitness* del nuevo individuo con el valor del *fitness* del

peor individuo de la población, y se selecciona el que mayor valor del *fitnees* tenga. En caso de que sea el nuevo individuo, esté pasará a formar parte de la población y el que pertenecía a la población, pero cuyo valor era menor que el del nuevo individuo, será extraído de ésta. Así se tiene la población preparada para ejecutar una nueva iteración. Este procedimiento se realizará tantas veces como indique el número de iteraciones.

Una vez realizadas todas las iteraciones el individuo con el mejor *fitness* de la población será la solución devuelta por el algoritmo.

7.6.2. Algoritmo genético II

A continuación se describe el segundo enfoque basado en algoritmos genéticos para la resolución del problema. Esta estrategia difiere considerablemente de la anterior, tanto en los tipos de operadores como en la representación y codificación de las soluciones.

- **Representación individual: Genotipo y Fenotipo**

Para representar el genotipo de un individuo usaremos un *string* o cadena. La longitud del *string* corresponde con el número de trabajos. Cada una de las posiciones del *string*, denotadas como genes, corresponden a un pedido y contienen el instante en el cual el pedido debería empezar. Este instante se define con respecto al instante ideal de comienzo del trabajo. La Figura 7.12 muestra el genotipo de un individuo en un problema con 5 pedidos. Para ese individuo, el trabajo 2 debería comenzar en el instante s_i-1, el trabajo 3 en el instante s_i+1 y el resto de los trabajos sobre su instante ideal de inicio.

0	-1	1	0	0

Figura 7.12: Genotipo del algoritmo genético II

Sea r el instante de comienzo del pedido i en un individuo, $r \in [a_i,b_i]$. El beneficio w_i del pedido i se obtiene como $w_i = \{ W_i + (r - s_i)w_i^- \text{ si } (r-s_i) < 0 \parallel W_i + (s_i - r)w_i^+ \text{ si } (r-s_i) \geq 0\}$

Es importante recalcar el hecho de que el valor de cada pedido en el individuo representa el instante en el que debiera comenzar, puesto que no existen garantías de admisibilidad para la realización de todos los pedidos con esos instantes de inicio. Por ello, cada genotipo representa el problema de seleccionar un subconjunto de pedidos que maximice el beneficio total, asumiendo esos instantes de comienzo para

los pedidos. Esta situación refleja el problema PDP con instantes fijos de entrega, es decir, el problema descrito en el Escenario I. Para resolver este problema, describimos un nuevo procedimiento heurístico que alcanza soluciones satisfactorias en un breve tiempo de computación.

En la primera fase del procedimiento heurístico se explota el hecho de que PDP puede ser modelado como un problema de flujo si la capacidad de producción es suficiente para procesar todos los pedidos en la planta en cualquier instante de tiempo. Haciendo uso de esta idea, no se consideran las restricciones asociadas con no procesar simultáneamente un número de pedidos mayor que la capacidad de producción de la planta.

La construcción de grafo dirigido *G* usado para resolver el problema está basada en la idea propuesta por [Hiraishi et al, 2002], descrita en la sección 3.4.2.1. Cada pedido *i* se representa en *G* por dos nodos, l_i y f_i que corresponden con los instantes de comienzo y fin de la fase de distribución, unidos por un arco con capacidad de una unidad y un coste $-w_i$. Además, existe un arco desde cada nodo f_i a todo nodo l_j siempre que $l_j \leq f_i$. Estos arcos tienen coste cero y capacidad uno, y representan la posibilidad de que un vehículo sirva el pedido *i* tras ser servir el pedido *j*. Un nodo inicial *s* se conecta a todos los nodos l_i y todos los nodos f_i se conectan a un nodo final *e* con coste cero y una unidad de capacidad.

Si inyectamos *V* unidades de flujo en el nodo *s* la solución óptima del problema de flujo a coste mínimo proporcionará la selección del subconjunto de pedidos con beneficio total máximo. Un pedido *i* será servido si y sólo si en la solución del problema de flujo una unidad de flujo circula a través del arco (l_i, f_i). La figura 7.13 muestra el grafo *G* correspondiente al genotipo mostrado en la figura 7.12 y referido al problema representado en la tabla 7.3, para un número de vehículos igual a 2.

	tp_i	td_i	s_i	a_i	b_i	W_i	w_i^-	w_i^+
Pedido 1	2	6	3	2	3	12	1	1
Pedido 2	2	11	4	3	4	20	1	1
Pedido 3	2	6	4	3	5	10	1	1
Pedido 4	3	6	13	12	14	13	2	1
Pedido 5	1	5	16	15	17	10	1	1

Tabla 7.3: Ejemplo del procedimiento heurístico

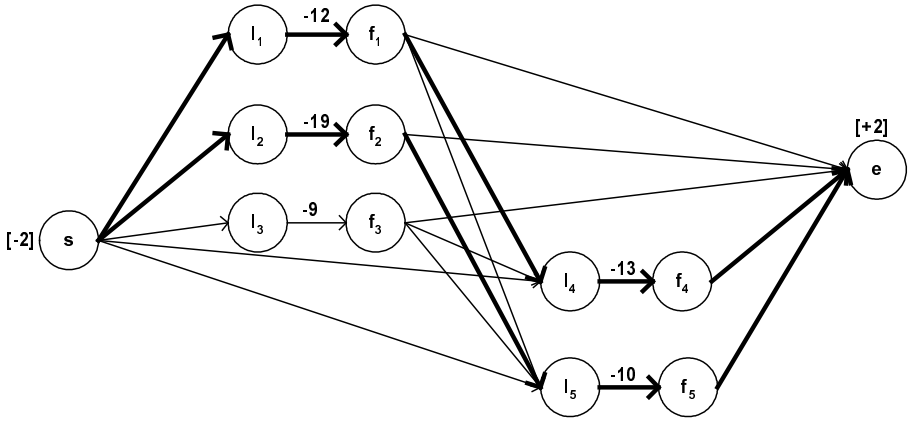


Figura 7.13: Grafo *G* correspondiente al genotipo de Figura 7.12.

Las líneas en grueso denotan el camino de coste mínimo. Los pedidos seleccionados son, por tanto, el pedido 1, 2, 4 y 5.

Una vez que se ha obtenido una solución para *G* se chequea la admisibilidad en la fase de producción de los pedidos seleccionados. Para dicha tarea, hacemos uso del lema de [Kroon, 1995] sobre el problema de la programación de trabajos fijos (FSP), el cual proporciona una condición necesaria y suficiente para la existencia de un plan admisible en la fase de producción de los pedidos seleccionados.

Sea P_s el conjunto de pedidos seleccionados. Un plan admisible que incluya todos los pedidos pertenecientes a P_s existe si y sólo si el máximo grado de simultaneidad de P_s en la fase de producción es menor o igual a la capacidad C de la planta.

Si suponemos que los pedidos son procesados en el intervalo de tiempo $[0,T]$, el máximo grado de simultaneidad se define como $L = \max\{L_t: 0 \leq t \leq T\}$ con $L_t = \{i \in P_s: s_i \leq t \leq l_i - 1\}$

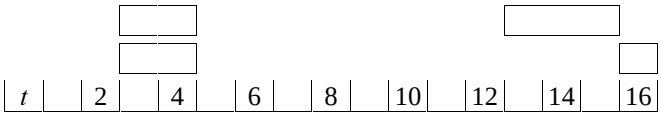


Figura 7.14: Fase de producción para los pedidos seleccionados en el grafo de figura 7.13

En la figura 7.14 las barras horizontales indican el tiempo de producción (tp_i) de los pedidos seleccionados previamente. Para este caso L es igual a 2. Es claro que sólo si $C \geq 2$ sería admisible procesar en la planta todos los pedidos seleccionados.

Si el máximo grado de simultaneidad de P_s excede de C, entonces la solución

proporcionada por el grafo no sería admisible. En ese caso, la heurística debe tratar de encontrar un subconjunto de pedidos con beneficio total máximo que puedan ser procesados con una capacidad de C unidades. Este problema llega a ser equivalente al problema Max. FSP (sección 3.4.1.1), el cual puede ser resuelto por un algoritmo de flujo a coste mínimo.

La construcción del grafo G' para la resolución de este problema puede ser descrita como sigue: El conjunto $R = \{r_d: d = 1, \ldots, D\}$ se usa para representar todos los tiempos de comienzo de los pedidos de P_s en orden cronológico. El conjunto de nodos del grafo es una correspondencia uno a uno con el conjunto R, además de un nodo final. Existe un arco desde cada nodo al siguiente con coste cero y capacidad ilimitada. Existe un arco también desde cada nodo al nodo correspondiente al primer pedido que podría producirse en la planta una vez finalizada la producción del pedido que representa al nodo que es origen del arco. Estos arcos transportan una unidad de flujo y tienen un coste igual a $-w_i$. En el primer nodo del grafo se inyectan C unidades de flujo, las cuales deben alcanzar el nodo final. Como ejemplo, la figura 7.15 muestra el grafo para los pedidos de la Figura 7.14, con capacidad $C = 1$.

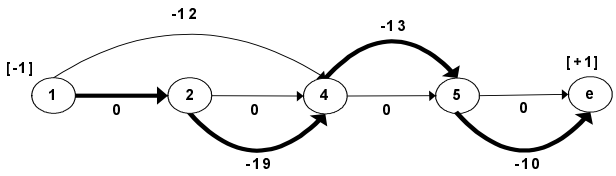


Figura 7.15: Grafo asociado al ejemplo de la figura 7.14

Una vez que se obtiene la solución óptima al problema de flujo a coste mínimo, sea E el conjunto de los pedidos pertenecientes a P_s que no han sido seleccionados ($E=\{1\}$ para el ejemplo). Para cada pedido $j \in E$ modificamos el grafo G, construido inicialmente, para prohibir que el pedido j pueda ser procesado. Para este fin, simplemente se necesita asignar capacidad cero a los arcos que unen l_j con f_j en G. Con este nuevo grafo G, todo el proceso descrito se repite de nuevo hasta encontrar una solución admisible, es decir, un conjunto de pedidos P_s que pueda ser procesado al completo en la planta de fabricación.

El pseudocódigo del proceso se muestra a continuación:

Crear Graph G
Repetir
  Cálculo de flujo a coste mínimo en G
  Sea P_s={i/ Flujo Arco $l_i \rightarrow f_i$ = 1}
  Crear grafo $G'(P_s)$
  Cálculo de flujo a coste mínimo en G'
  E = {$j \in P_s$: Flujo arco (j, Posterior(j)) = 0}
  Si $E = \varnothing$ **entonces**
    Admisibilidad = *true*
  si no
    Admisibilidad = *false*
    Capacidad arcos (l_j, f_j) = 0 en G, $j \in E$
  Fin Si
Hasta Admisibilidad = *true*

- **Selección y reproducción de la población**

En el algoritmo hemos usado la técnica *steady-state*, reemplazando un individuo en cada iteración. Por tanto, en cada iteración un nuevo individuo se genera usando unos determinados operadores. En cada iteración un operador será seleccionado para generar un nuevo individuo. Para esta selección, cada operador tendrá una probabilidad de ser elegido.

La selección del individuo que va ser eliminado se realiza a través de un ranking exponencial de los individuos dentro de la población. El ranking exponencial asigna una probabilidad p de ser eliminado al peor individuo de la población, según su fitness. Si el individuo no es seleccionado, entonces el siguiente al último tendrá también una probabilidad p, y así sucesivamente. El valor establecido para esa probabilidad fue de 0.3.

- **Operadores de cruce y mutación**

El operador de cruce es el operador más importante para generar nuevos individuos, es decir, nuevos puntos de búsqueda. Este operador emplea dos individuos denominados padres y produce un nuevo individuo llamado *offspring* o hijo, mediante una manipulación de partes de los padres.

El operador de cruce usado en este algoritmo ha sido el siguiente: Se seleccionan aleatoriamente dos padres *p1* y *p2*. El hijo se construye con la siguiente regla:

Sea $p1(i)$ el gen i en $p1$. Para cada i desde 1 hasta n, si $p1(i) = p2(i)$ entonces $hijo(i) = p1(i)$. En otro caso hijo(i) es un valor aleatorio en el intervalo [$p1(i)$, $p2(i)$].

En la Figura 7.16 se presenta un ejemplo de este operador.

p1	2	-1	0	1	0
p2	1	-1	1	-1	0
Hijo	Aleatorio[1,2]	-1	Aleatorio[0,1]	Aleatorio[-1,1]	0

Figura 7.16: Operador de Cruce

También se ha empleado un operador estándar de mutación que aleatoriamente selecciona un individuo y aleatoriamente también elige un nuevo valor para alguna de sus posiciones. Este operador ayuda al algoritmo a mantener la diversidad en la población para evitar una convergencia prematura.

- **Valores de los parámetros del algoritmo**

Tras analizar empíricamente diferentes valores para los parámetros del algoritmo genético II, la tabla 7.4 recoge aquellos que dieron lugar a un mejor comportamiento del algoritmo:

Parámetro	Valor
Tamaño de la población	20
Población inicial	Aleatoria
Probabilidad de Cruce	0.6
Probabilidad *p* para el ranking	0.3
Número de iteraciones	1000

Tabla 7.4: Valores de los parámetros del algoritmo

7.7. RESULTADOS COMPUTACIONALES

La primera etapa de nuestra experimentación fue la construcción de una batería de problemas. La dificultad para obtener soluciones óptimas en este escenario, limita el tamaño de los problemas generados. No obstante, problemas de hasta 75 pedidos se consideran suficientes para extraer conclusiones acerca de los métodos heurísticos diseñados.

Posteriormente, en la sección 7.7.2 se realiza un análisis del comportamiento del método exacto. Finalmente, los experimentos realizados sobre los diferentes métodos heurísticos se presentan en la sección 7.7.3.

7.7.1. Generación de problemas

Los tamaños de los problemas usados fueron 20, 30, 40, 50 y 75 pedidos. Diez instancias de cada tamaño fueron generadas aleatoriamente.

Como parámetros principales de la generación se consideró el promedio de solapamiento de los pedidos en las fases de producción (L_m^P) y distribución (L_m^D). El cálculo de los mismos se describe en el Anexo I. Estos parámetros proporcionan una visión clara del nivel de dificultad en la resolución de los problemas. Los valores obtenidos están recogidos en la siguiente tabla:

n	L_m^P	L_m^D
20	1.51	4.97
30	1.54	5.45
40	1.49	5.03
50	1.58	5.94
75	2.09	7.80

Tabla 7.5: Promedios en el grado de simultaneidad

Estos datos van referidos a cantidades promedio por instante en el horizonte de planificación. Sin embargo, en algunos problemas el grado de solapamiento sube por encima de 12 pedidos en la fase de distribución y de 4 pedidos en la fase de producción, en alguno de los instantes del horizonte de planificación.

El horizonte de planificación fue considerado dependiente del número de pedidos del problema de forma que se mantuvieran, en función de los tiempos de actividad de los pedidos, los promedios de solapamiento antes apuntados. Según ello, el horizonte estuvo de acuerdo a los siguientes intervalos: [1,55] para 20 pedidos; [1,75] para 30 pedidos; [1,95] para 40 pedidos; [1,115] para 50 pedidos y [1,155] para 75 pedidos. Los tiempos de proceso en planta, tiempos de viaje, amplitud de las ventanas temporales, valores de los pedidos y penalizaciones por retraso y adelanto, fueron aleatoriamente obtenidos dentro de los intervalos que muestra la tabla 7.6.

tp_i	ti_i	tu_i	tr_i	b_i-a_i	W_i	w^-_i	w^+_I
[1,5]	[2,10]	[1,2]	[1,10]	[1,5]	[30,100]	[0,2]	[0,2]

Tabla 7.6: Intervalos para valores de datos

Para permitir diferentes niveles con respecto a la capacidad C y al número de vehículos V, los siguientes pares de valores (C,V) fueron considerados en la resolución de los problemas: (1,2), (2,2), (2,3), (1,3) y (1,4).

Los tiempos vienen expresados en segundos sobre un Pentium III 850 Mhz.

7.7.2. Comportamiento del método exacto

(C,V)	n	Número de nodos	Número de arcos	Tiempo de computación	% Pedidos servidos
(1, 2)	20	3608.7	26833.8	35.5	33.0
	30	5547.7	114149.3	40.6	32.7
	40	8105.4	231344.7	97.7	34.0
	50	10365.0	312296.4	178.5	31.2
	75	27280.4	1152220.1	2821.3	25.6
(2, 2)	20	6634.2	45373.2	23.9	36.0
	30	11208.1	217060.8	79.6	35.3
	40	14564.9	406417.6	142.0	35.8
	50	18375.3	520874.5	210.7	32.6
	75	60609.8	2324479.4	3134.4	26.7
(1, 3)	20	20175.4	160669.3	336.7	44.0
	30	27513.1	610562.6	763.7	45.0
	40	45654.3	1417960.1	3112.2	44.3
	50	52576.0	1742924.3	4580.1	43.6
	75	173136.2	7462447.4	20214.2	35.3
(2, 3)	20	84864.0	613873.3	1495.7	49.5
	30	122914.7	2617693.6	6911.5	48.3
	40	158233.5	4946806.3	12996.6	48.0
	50	179093.5	5673009.3	15617.0	46.2
(1, 4)	20	51134.2	418209.8	713.1	52.0
	30	66113.7	1500364.5	4309.9	53.7
	40	110846.4	3682355.6	9122.2	52.2
	50	130604.0	4467145.7	14127.1	52.6

Tabla 7.7: Sumario de resultados del método exacto

La tabla anterior resume el comportamiento del método exacto en la resolución de problemas. Se muestran numero de nodos, número de arcos, tiempos de computación y porcentaje de pedidos servidos, respecto a n, C y V. Todos los datos son cantidades medias respecto a las 10 instancias resueltas en cada categoría.

Las instancias con 75 pedidos no pudieron ser resueltas con recursos (C,V) = (2,3); (1,4) debido al excesivo consumo de memoria del algoritmo.

A continuación se presentan los datos de forma agregada respecto a n. Las leyendas de los gráficos son las reflejadas en la figura 7.16.

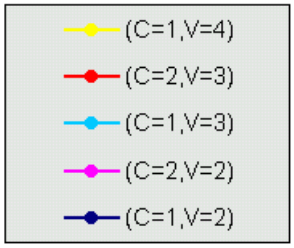
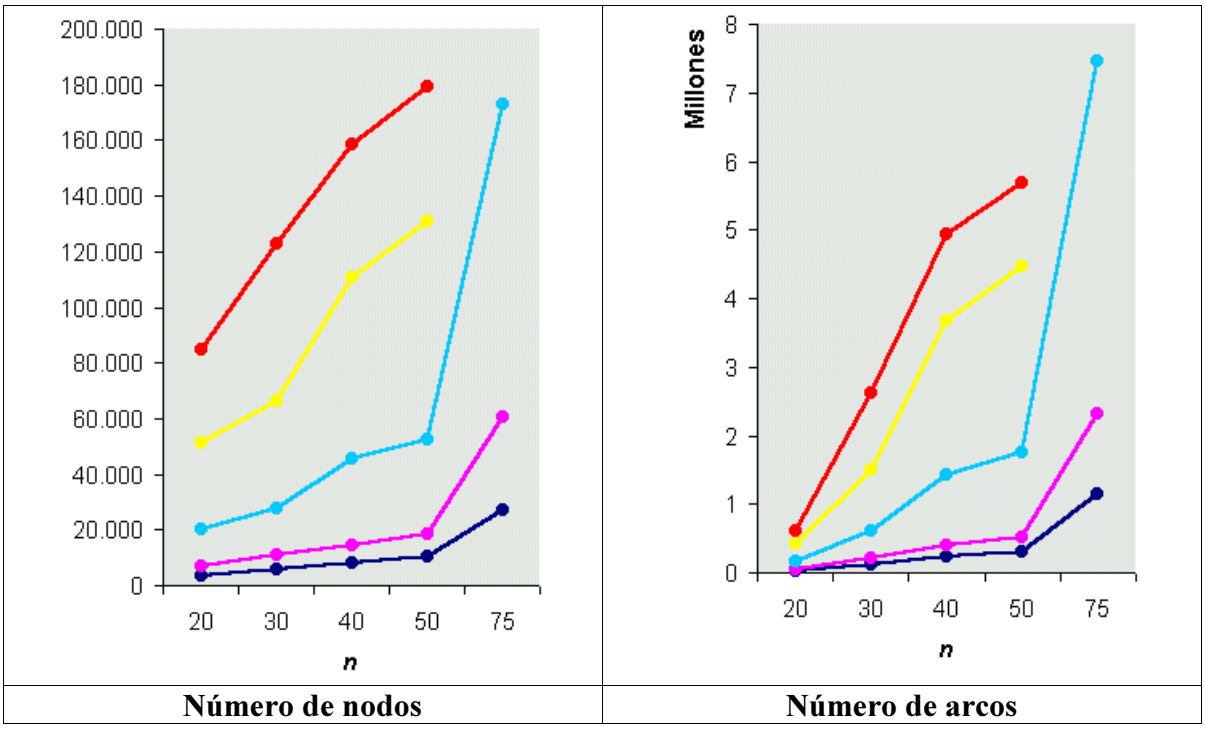


Figura 7.16: Leyendas de los gráficos



Número de nodos **Número de arcos**

Figura 7.17: Número de nodos y arcos

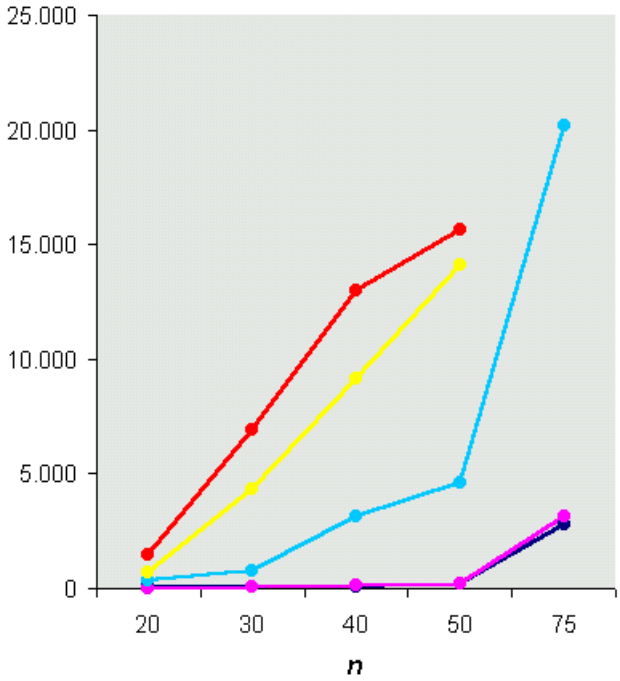


Figura 7.18: Tiempos de computación

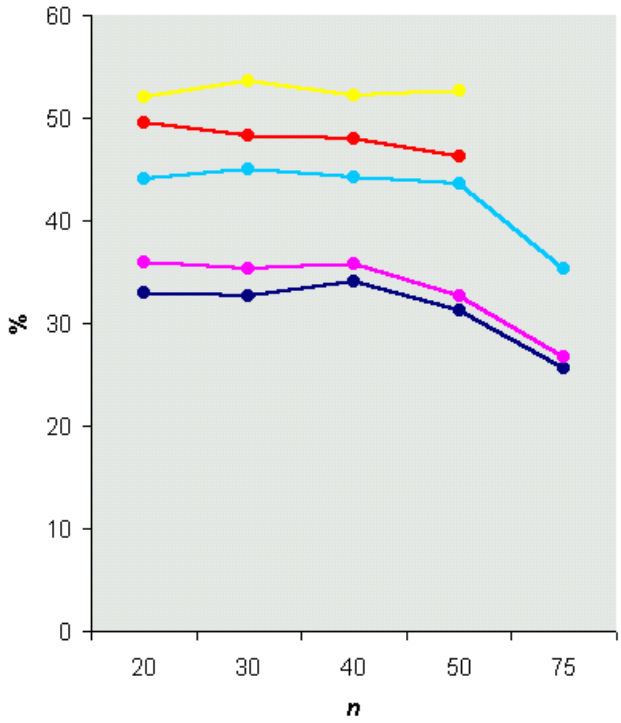


Figura 7.19: Porcentaje de pedidos servidos

Tal como se aprecia en las figuras anteriores, el crecimiento del número de arcos, nodos y tiempos de computación, es exponencial con el tamaño del problema y el número de recursos, haciendo inviable problemas con más de 75 pedidos o 5 recursos totales.

7.7.3. Comportamiento de los métodos heurísticos

En este apartado se presentan los resultados de la búsqueda tabú y de los algoritmos genéticos. Comenzamos describiendo un estudio de sensibilidad realizado para el algoritmo genético I. Tras este análisis se realiza una comparativa de los dos algoritmos genéticos. La mejor implementación, según este análisis, se compara finalmente con la búsqueda tabú.

7.7.3.1. Análisis de sensibilidad del algoritmo genético I

En esta sección se analiza el comportamiento del Algoritmo Genético I ante diferentes valores de sus parámetros. Estos parámetros son:

- Probabilidad de cruce y mutación
- Tamaño de la población

- Individuos iniciales
- Número de iteraciones

Para el estudio se ha escogido un conjunto de 10 problemas de 25 pedidos. Los valores para *C* y *V* han sido: (1,2), (2,2) y (2,3).

❑ Probabilidad de cruce y mutación

El estudio se ha realizado con un valor fijo del resto de parámetros, los cuales han sido:

- Tamaño de la población: 20
- Individuos iniciales obtenidos aleatoriamente
- Número de iteraciones: 2000

A continuación se muestra el error medio para 5 valores de la probabilidad de cruce.

Probabilidad de cruce	Error medio
0.3	4.25
0.4	4.93
0.5	3.92
0.6	3.90
0.7	4.71

Tabla 7.8: Error medio según probabilidad de cruce

La probabilidad que mejor resultado proporcionó fue 0.6 para el operador de cruce y, por tanto, 0.4 para el operador de mutación.

❑ Tamaño de la población

Tres valores fueron analizados para el tamaño de la población. El valor del resto de los parámetros del algoritmo para el análisis fue:

- Probabilidad de cruce: 0.6
- Individuos iniciales obtenidos aleatoriamente
- Número de iteraciones: 2000

Nº de individuos	Error medio
15	4.66
20	3.92
25	4.71

Tabla 7.9: Error medio en función del número de individuos de la población

- **Población inicial**

Para el estudio de la población inicial, se analizaron las 3 técnicas comentadas en la sección 6.4.1:

- Generación aleatoria de individuos
- Generación basada en una elección avariciosa
- Generación híbrida: 50% obtenidos aleatoriamente y 50% obtenidos mediante elección *greedy*.

El resto de los parámetros fueron:

- Probabilidad de cruce: 0.6
- Tamaño de la población: 20
- Número de iteraciones: 1000

Los resultados fueron:

Población Inicial	Error medio
Aleatoria	4.88
Híbrida	4.07
Avariciosa	4.35

Tabla 7.10: Error según población inicial

A la vista de los resultados, se aconseja el uso de una población inicial obtenida tanto de una forma aleatoria como avariciosa.

- **Número de iteraciones**

En general, las soluciones mejoran al aumentar el número de iteraciones, puesto que, aumenta la probabilidad de explorar nuevas soluciones. Con el estudio del número de iteraciones tratamos de examinar el aumento de la calidad de las soluciones con relación al aumento del número de iteraciones, buscando un número de iteraciones a partir del cual el incremento no signifique un aumento notable de la calidad de las soluciones. Hay que tener en cuenta, que tan importante es la calidad de las soluciones como el tiempo de computación para llegar a ellas.

Se han estudiado 500, 1000, 1500 y 2000 iteraciones. Los errores medios obtenidos se muestran en la siguiente tabla:

Número de iteraciones	Error medio	Tiempo medio
500	5.52	36.2
1000	4.25	66.2
1500	3.97	91.5
2000	3.92	119.6

Tabla 7.11: Error y tiempo medio según número de iteraciones

El salto más significativo en el error se produce de 500 a 1000 iteraciones.

7.7.3.2. Comparación de los algoritmos genéticos

A continuación se presentan los resultados obtenidos con los dos algoritmos genéticos, respecto a los problemas de 20, 30, 40 y 50 pedidos. A partir del estudio de sensibilidad realizado sobre el algoritmo I y tras un análisis similar del algoritmo II, los valores de los parámetros de los algoritmos fueron los siguientes:

	ALGORITMO GENÉTICO I	ALGORITMO GENÉTICO II
Población Inicial	Híbrida	Aleatoria
Tamaño de la población	20	20
Probabilidad Cruce	0.6	0.6
Tipo Reproducción	*Steady-state*	*Steady-state*
Eliminación	Peor fitness	Ranking exponencial (p=0.3)
Número de iteraciones	1000	1000

Tabla 7.12: Parámetros de los algoritmos

Los resultados obtenidos tras la ejecución de los dos algoritmos fueron los siguientes:

		Algoritmo Genético I			Algoritmo Genético II		
(*C*,*V*)	***n***	**Error**	**Nº Óptimos**	**Tiempo**	**Error (%)**	**Nº Óptimos**	**Tiempo**
(1, 2)	20	2.89	3	57.1	0.28	9	670.0
	30	2.02	4	81.2	0.24	7	706.0
	40	4.95	0	100.5	0.55	2	709.0
	50	5.21	0	133.8	0.55	3	1017.0
(2, 2)	20	2.40	3	57.4	0.00	10	560.8
	30	2.58	3	80.6	0.37	6	561.9
	40	4.52	0	102.4	0.54	2	569.1
	50	4.52	1	134.8	0.36	4	1041.9
(2, 3)	20	3.94	2	63.2	0.05	7	614.1
	30	4.09	2	93.6	0.24	7	618.2
	40	5.13	0	116.8	0.39	3	659.1
	50	5.99	0	139.4	0.97	0	1653.0

Tabla 7.13: Sumario de resultados de los algoritmos genéticos

Comentarios

A la vista de los resultados se pueden extraer las siguientes conclusiones:

- El algoritmo genético II, cuya decodificación incidía en las características del problema PDP, presenta unos resultados notablemente mejores respecto al error medio y al número de óptimos.

- Sin embargo, los tiempos de computación son mayores en el algoritmo II. La razón es que en cada iteración del algoritmo, para el cálculo de un nuevo individuo se resuelve, al menos, un algoritmo de flujo a coste mínimo. En cambio, el cálculo de un individuo nuevo en el algoritmo genético I es un proceso sencillo.

7.7.3.3. Análisis de resultados de la búsqueda tabú

Para el análisis de la búsqueda tabú, se presentan en primer lugar los resultados proporcionados por el algoritmo. Posteriormente se presentan resultados comparativos del error medio y tiempos de computación respecto al algoritmo genético II. El algoritmo de búsqueda tabú realiza en todas las ejecuciones 5000 iteraciones.

(*C*,*V*)	*n*	Error medio (%)	Nº de óptimos	Tiempo medio
(1, 2)	20	0.00	10	7.5
	30	0.13	7	11.1
	40	0.21	7	14.1
	50	0.12	5	17.0
	75	1.44	2	25
(2, 2)	20	0.58	6	8
	30	0.88	7	11
	40	0.35	6	15
	50	0.84	5	19
	75	1.94	3	30
(1, 3)	20	0.29	9	10
	30	0.30	9	14
	40	0.32	4	18
	50	0.23	3	21
	75	0.59	1	33
(2, 3)	20	0.06	9	11.2
	30	0.30	7	16.6
	40	0.50	1	20
	50	0.62	1	37
(1, 4)	20	0.38	9	11
	30	0.42	7	15
	40	0.61	3	18
	50	0.58	4	42

Tabla 7.14: Resultados de la búsqueda tabú

7.7.3.4. Comparación de búsqueda tabú y algoritmo genético II

A continuación se presentan gráficas comparativas del error, número de óptimos y tiempo de computación de los dos métodos:

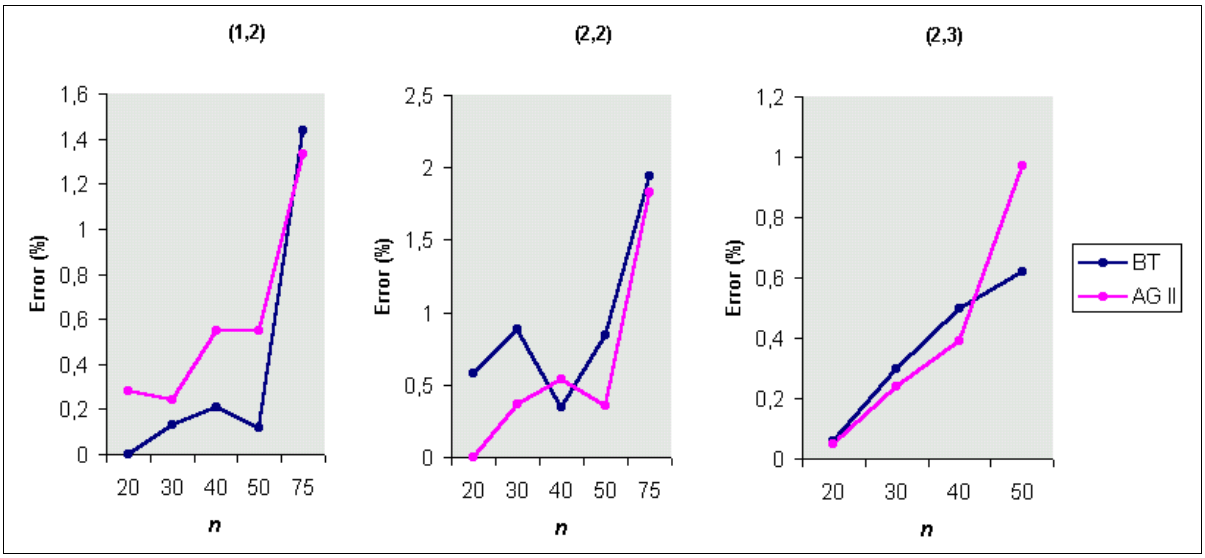


Figura 7.20: Error medio

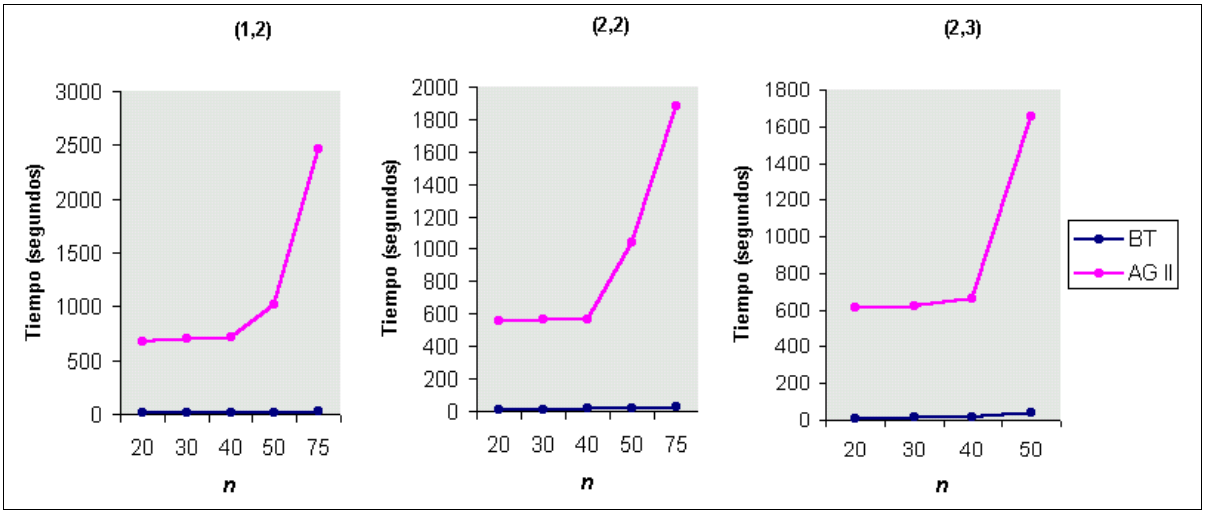


Figura 7.21: Tiempo medio

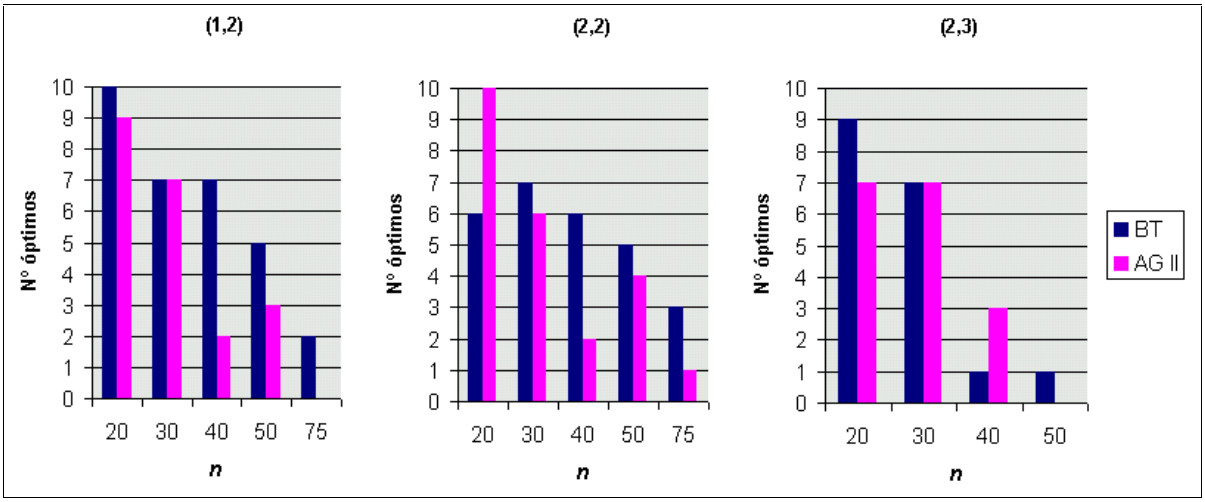


Figura 7.22: Número de óptimos

Los datos agregados por tamaño se presentan en la siguiente tabla:

	Búsqueda tabú						Algoritmo Genético II					
Tamaño	**20**	**30**	**40**	**50**	**75**	**Total**	**20**	**30**	**40**	**50**	**75**	**Total**
Error	0.21	0.44	0.35	0.53	1.69	**0.64**	0.11	0.28	0.49	0.63	1.58	**0.62**
Nº Opt.	25	21	14	11	5	**76**	26	20	7	7	1	**61**
Tiempo	8.9	12.9	16.4	24.3	27.5	**18.0**	615.0	628.7	645.7	1237.3	2170.0	**1059.3**

Tabla 7.15: Datos agregados de BT y AG II por tamaño de problema

Comentarios

A la vista de los datos se pueden extraer las siguientes conclusiones:

- Búsqueda tabú encontró soluciones óptimas en 76 de los 230 problemas resueltos. El algoritmo genético II mostró un peor comportamiento en este aspecto, con 61 éxitos.

- El error medio fue prácticamente el mismo en los dos métodos, con una ligera mejora del algoritmo genético II. Hay que tener en cuenta que el algoritmo genético se corrió con 1000 iteraciones, por 5000 de la búsqueda tabú. El hecho de no ajustar el número de iteraciones se debe al tiempo de computación empleado por el algoritmo genético, el cual es sensiblemente peor al de la búsqueda tabú, a pesar de usar menor número de iteraciones. Sin embargo, la igualdad en el error con un número bastante menor de iteraciones pone de manifiesto la mayor convergencia del algoritmo genético.

Capítulo 8

Escenario III: El problema PDP con instantes fijos de entrega y varias plantas de producción

Índice

8.1. INTRODUCCIÓN

El Escenario III se ocupa del problema asociado a la planificación conjunta de la fabricación y distribución de pedidos para el caso de múltiples plantas de producción e instantes fijos de entregas. Como en los escenarios anteriores, se busca la selección de pedidos que maximice el beneficio de la planificación.

Este escenario se enfoca como una derivación de los escenarios anteriores, y más en concreto del escenario I. Por ello, nos ayudaremos de parte del trabajo realizado en los escenarios previos.

El contenido del capítulo es el siguiente:

- Descripción del problema: Introducimos el problema PDP con varias plantas, apuntando las características generales del proceso más las que se agregan al problema en este escenario.

- Formulación: Se presenta un modelo de programación entera que maximiza el beneficio de los pedidos servidos.

- Planteamiento de un caso particular: En la sección 8.4 consideramos el caso en el que la capacidad de producción de cada planta es mayor o igual al número de vehículos disponibles. Para este caso, presentamos un método óptimo que puede ser considerado como un procedimiento para obtener una cota superior de la solución óptima para el caso general. Este método será usado como base para la heurística descrita en la sección para el caso general.

- Resolución del problema en el caso general: Como herramientas de resolución del problema, se describen, en la sección 8.5, tanto un método exacto como un procedimiento heurístico basado en un problema de flujo a coste mínimo.

- Finalmente, en la sección 8.6 se presentan los resultados computacionales.

8.2. DESCRIPCIÓN DEL PROBLEMA

En este escenario se considera el problema de planificar un conjunto dado de pedidos, de forma que los pedidos seleccionados para ser servidos deben ser preparados en una cualquiera de las plantas de producción existentes, e inmediatamente distribuidos al lugar del cliente. Existe un conjunto de m plantas de producción K={1...m}, cada una con una capacidad de producción limitada C_k. Cada pedido i puede únicamente ser preparado en un subconjunto K_i del conjunto de plantas K.

Existe un número fijo de vehículos de idénticas características. Una vez que un pedido se prepara en la planta, se distribuye sin esperas al lugar de destino. Asumimos que el tamaño de cada pedido es inferior a la capacidad de un vehículo y que sólo un pedido puede ser distribuido en cada viaje que realice un vehículo. Cuando un pedido es descargado en el destino, el vehículo retorna a cualquiera de las plantas existentes y queda disponible para el reparto de otro pedido. Inicialmente, cada vehículo se encuentra en una planta específica. Consideraremos v_k como el número de vehículos inicialmente en la planta k.

El número de pedidos, sus tiempos de proceso en planta y los tiempos de viaje desde las localizaciones de destino a cualquiera de las plantas son datos conocidos y fijos. De ese modo, con cada pedido i, del conjunto de pedidos P={1...n}, se asocia una fecha de entrega e_i, un tiempo ti_{ik} desde cada planta k a su localización, así como un tiempo de viaje tr_{ik} desde la localización hasta cada planta. Además de estos datos, también existe un tiempo de producción tp_i independiente de la planta, y un tiempo de descarga tu_i. Al igual que en el Escenario I, sólo se admite la entrega del pedido i en el instante e_i. La figura 8.1 muestra el proceso gráfico asociado a la actividad de un pedido.

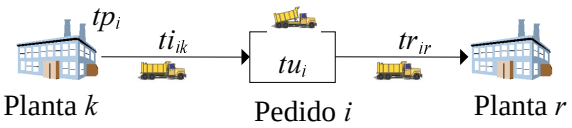


Figura 8.1: Representación gráfica

A partir de esos datos, podemos considerar $s_{ik} = e_i - ti_{ik} - tp_i$ como el instante de comienzo de la producción del pedido en la planta k, $l_{ik} = e_i - ti_{ik}$ como el instante de comienzo de la distribución del pedido desde la planta k, y $f_{ir} = e_i + tu_i + tr_{ir}$ el instante de llegada del vehículo a la planta r. La figura 8.2 presenta estos datos.

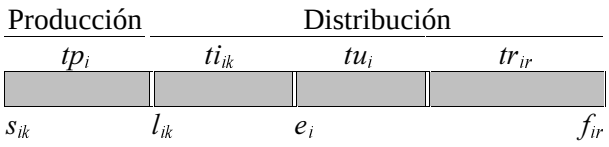


Figura 8.2: Actividad de un pedido en el Escenario III

Cada pedido tiene un valor positivo w_i reflejando el beneficio asociado con ser servido. Los costes de distribución son proporcionales al tiempo de distribución. Los costes de producción son los mismos para todas las plantas y por tanto no se consideran.

El objetivo en el problema será maximizar el beneficio asociado a la selección de pedidos realizada, considerando tanto el valor de los pedidos servidos como los costes de transporte a que han dado lugar.

8.3. FORMULACIÓN DEL PROBLEMA

Para formular el problema introducimos una escala de tiempo discreto diviendo el tiempo en T periodos, con $t = 1...T$. Para cada pedido $i \in P$ y cada planta $k \in K_i$ en todo instante de tiempo $t \in [s_{ik}, l_{ik} - 1]$ definimos $S(t,k) = \{i \in P : s_{ik} \leq t \ \& \ l_{ik} > t\}$ como el conjunto de pedidos cuya fase de producción en la planta k se produce en el instante t. Sea d_{ik} el coste de transportar el pedido i desde la planta k, $d_{ik} = ti_{ik}$, y d'_{ik} el coste de la vuelta del vehículo a la planta desde la localización del pedido i, $d'_{ik} = tr_{ik}$.

❑ Variables

Nuestro modelo incorpora las siguientes variables de decisión:

- $x_{ik} = 1$ si el pedido i se procesa en la planta k; 0 en otro caso. ($\forall i \in P$, $k \in K_i$)

- $z_{ik} = 1$ si el vehículo que sirve el pedido i vuelve a la planta k; 0 en otro caso. ($\forall i \in P$ y $k \in K$)

- y_{tk} = número de vehículos disponibles en el instante t en la planta k, $t \in \{1..T\}$.

- **Restricciones**

1. Cada pedido puede ser procesado a lo sumo en una planta:

$$\sum_{k \in K_i} x_{ik} \leq 1 \quad \forall i \in P$$

2. Tras la descarga de un pedido, una vez servido, el vehículo debe retornar a una cualquiera de las plantas:

$$\sum_{k \in K_i} x_{ik} - \sum_{k=1}^{K} z_{ik} = 0 \quad \forall i \in P$$

3. Número de vehículos iniciales en cada planta:

$$y_{0k} = v_k \quad \forall k \in K$$

4. Cómputo del número de vehículos disponibles en cada planta en el instante *t*:

$$y_{tk} = y_{0k} - \sum_{i / l_{ik} < t} x_{ik} + \sum_{i / f_{ik} \leq t} z_{ik} \qquad \forall t \in \{1..T\}; \forall k \in K$$

5. Deben existir vehículos disponibles en el instante l_{ik} para poder procesar el pedido *i* en la planta *k*.

$$\sum_{i / l_{ik} = t} x_{ik} \leq y_{tk} \qquad \forall t \in \{1..T\}; \forall k \in K$$

6. No más de C_k pedidos pueden ser procesados al mismo tiempo, en cualquier instante, en la planta *k*.

$$\sum_{i \in S(t,k)} x_{ik} \leq C_k \quad \forall k \in K, \forall t \in [1 \ldots T]$$

El modelo para una función objetivo que maximiza el beneficio de la planificación es el siguiente:

$$Max \sum_{i=1}^{n} \sum_{k \in K_i} w_i x_{ik} - \left(\sum_{i=1}^{n} \sum_{k \in K_i} d_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{k=1}^{K} d'_{ik} z_{ik} \right)$$

sujeto a

$$\sum_{k \in K_i} x_{ik} \leq 1 \quad \forall i \in P \tag{1}$$

$$\sum_{k \in K_i} x_{ik} - \sum_{k=1}^{K} z_{ik} = 0 \quad \forall i \in P \tag{2}$$

$$y_{0k} = v_k \quad \forall k \in K \tag{3}$$

$$y_{tk} = y_{0k} - \sum_{i / l_{ik} < t} x_{ik} + \sum_{i / f_{ik} \leq t} z_{ik} \qquad \forall t \in \{1..T\}; \forall k \in K \tag{4}$$

$$\sum_{i / l_{ik} = t} x_{ik} \leq y_{tk} \qquad \forall t \in \{1..T\}; \forall k \in K \tag{5}$$

$$\sum_{i \in S(t,k)} x_{ik} \leq C_k \quad \forall k \in K, \forall t \in [1 \ldots T] \tag{6}$$

$$x_{ik}, z_{ik} \in \{0,1\} \quad y_{tk} \text{ Enteras}$$

8.4. NÚMERO DE VEHÍCULOS MENOR O IGUAL A LA CAPACIDAD DE PRODUCCIÓN DE CADA PLANTA

Podemos definir el número total de vehículos en el problema como $V = \sum_{k=1}^{m} v_k$

Si $V \leq C_k$, $\forall k \in K$ las restricciones (6) del modelo no imponen ninguna limitación. El problema cobra entonces un cierto interés puesto que este caso se reduce a encontrar el subconjunto de pedidos con beneficio total máximo que puedan ser servidos por los vehículos disponibles. Como V no excede la capacidad de producción en cada planta, eso significa que existe capacidad suficiente en cualquier planta para procesar todos los pedidos que pudieran estar siendo distribuidos en cualquier instante.

El problema en este caso particular adquiere un cierto componente real, ya que normalmente es el proceso de distribución en el que impone las limitaciones en actividades de este tipo.

Este caso particular puede ser resuelto en tiempo polinomial por un algoritmo de flujo a coste mínimo. Además, este mismo procedimiento pude ser usado para obtener una cota superior del valor de la solución óptima en el caso general.

La construcción del grafo directo G que usamos para resolver el problema puede ser descrito como sigue. Por cada planta existe un nodo en G. Representamos esos nodos como $\{n_1 \ldots n_k \ldots n_m\}$. Para cada pedido i creamos un subgrafo G_i con la siguiente estructura: Nodos n^i_{ok}, k=1...m, definen cada una de las plantas donde el pedido i pudiera ser procesado, con independencia de que $k \notin K_i$. Nodos n_{is} y n_{ie} definen el comienzo y fin de la descarga del pedido. Los nodos n^i_{rk} k=1...m representan las plantas de retorno para el vehículo que sirve el pedido i.

Los costes de transporte desde cada planta al lugar de destino del pedido i se representan como costes en los arcos que unen los nodos n^i_{ok}, asociados a las plantas, con los nodos n_{is} de comienzo de descarga del pedido. Los costes de retorno son también representados en G_i de forma similar, es decir, en los arcos (n_{ie} , n^i_{rk}). El valor del pedido i se representa como un coste negativo $-w_i$ en el arco que define la descarga (n_{is} , n_{ie}). Puesto que un pedido es procesado a lo sumo una vez, la capacidad para cada arco en G_i es 1. Sin embargo, cuando una planta k no pertenece a K_i, la capacidad del arco que representa este viaje es cero.

Para cada nodo planta n_k existe una arco desde n_k a n^i_{ok}, i = 1...n, con capacidad uno y coste cero. Para permitir que un vehículo que retorna a una planta, quede disponible para servir otros pedidos desde esa planta, se aplica la siguiente regla:

$$\{ \text{Crear arco desde } n^i_{rk} \text{ a } n^j_{dk} \text{ si } f_{ik} \leq s_{jk} \}$$

Dicha regla crea nuevos arcos en G que reflejen esta posibilidad. El número de arcos que pueden ser creados con esta regla es menor o igual a $m(n^2\text{-}n)/2$.

Finalmente, se conectan todos los nodos n^i_{rk} i = 1...n, k = 1...m, a un nodo final e. Estos arcos tienen coste nulo y pueden transportar una unidad de flujo. Un grafo genérico G se muestra en la siguiente figura:

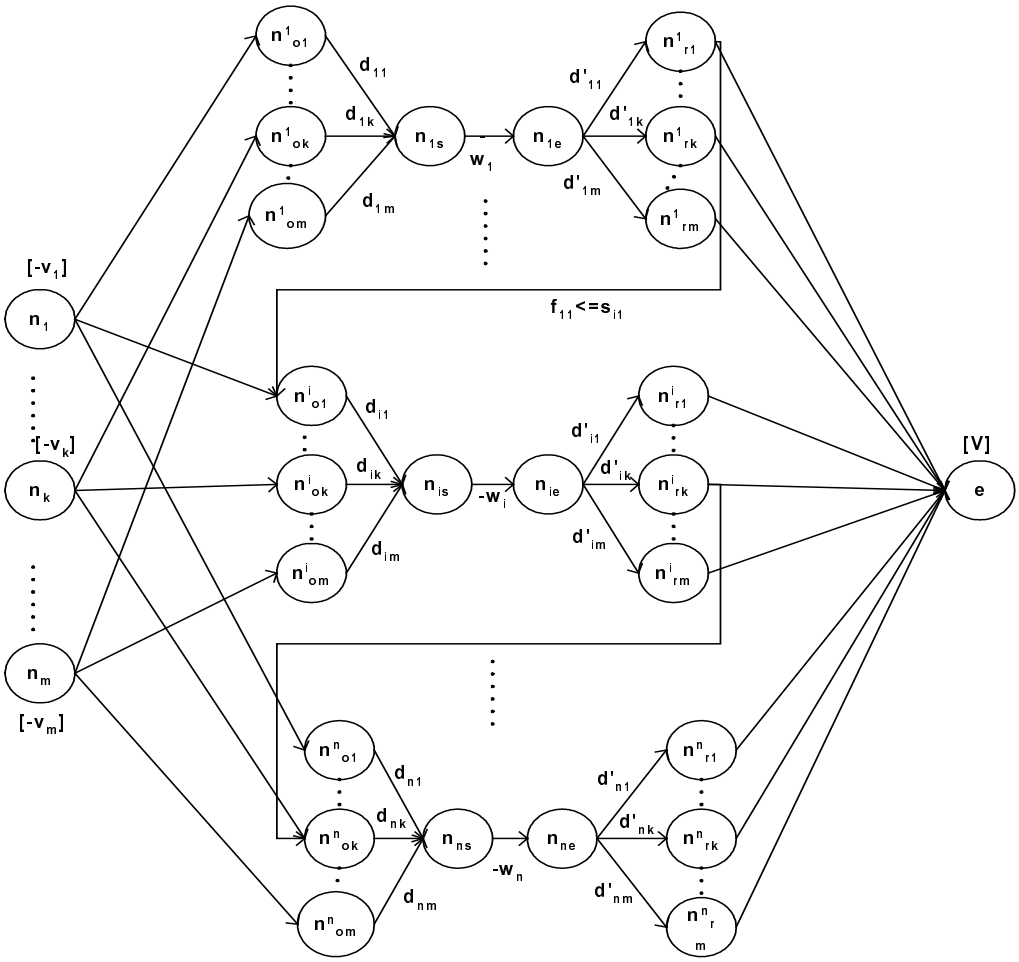


Figura 8.3: Gráfico genérico *G*

El número de nodos en *G* es igual a $m + 2m + 2n$. El número de arcos es menor o igual a $3mn + n + m(n^2-n)/2$. Si inyectamos v_k (es decir, el número de vehículos iniciales en la planta k) unidades de flujo en el nodo n_k, $k = 1...m$, el camino con coste mínimo en *G* proporcionará los pedidos a ser procesados que maximizan el beneficio. Obviamente, un pedido i es procesado si una unidad de flujo circula a través del subgrafo G_i.

Este procedimiento puede ser usado como método de obtención de una cota superior de la solución óptima para problemas en los que no existan las limitaciones impuestas en este caso particular, es decir, en donde no se cumpla que $V \leq C_k$ $k = 1...m$.

Al igual que en problemas de flujo descritos en otros capítulos, la solución óptima al problema se obtiene con el uso de RELAX IV, en particular la implementación C++ disponible en [Frangioni, 2001].

Como ilustración del método, se presenta un pequeño ejemplo con 3 pedidos y 2

plantas (Tabla 8.1) cuyo grafo aparece en figura 8.4.

P	s_{i1}	s_{i2}	l_{i1}	l_{i2}	f_{i1}	f_{i2}
1	2	3	3	4	10	11
2	6	4	7	5	12	14
3	10	9	11	10	13	13

Tabla 8.1: Ejemplo para el caso particular

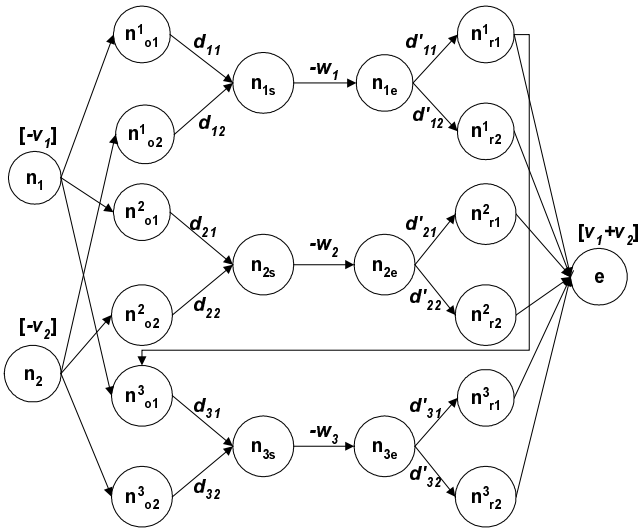


Figura 8.4. Grafo *G* para el ejemplo de Tabla 8.1

8.5. DOS ENFOQUES PARA RESOLVER EL PROBLEMA

En esta sección proponemos dos estrategias para resolver el problema en el caso general. La primera estrategia es un método de resolución exacto. La razón principal para desarrollar este método es que sus resultados pueden guiar las conclusiones sobre el método heurístico descrito en la sección 8.5.2.

8.5.1. Un método exacto basado en grafos

El enfoque que presentamos es similar al método exacto propuesto en el Escenario IA del problema, aunque con un mayor grado de complejidad. El método construye un grafo *G* que colecciona todas las soluciones admisibles del problema a través de un método de evaluación de estados admisibles en la planificación de los pedidos. El camino más largo desde el nodo de salida al nodo final en *G* es la solución óptima del problema.

El proceso puede ser descrito como sigue: Consideremos los eventos correspondientes al comienzo del procesamiento de cada pedido *i* en cada planta *k*, $k \in K_i$. Sea *E* el número total de esos eventos y sea el conjunto $\{t_e / e = 1,\ldots, E\}$ usado para representar esos eventos en orden cronológico. Es decir, $\{t_e / e = 1\ldots E\} = \{s_{ik} : i = 1\ldots n, k = 1\ldots m, k \in K_i\}$ y $t_{e-1} \leq t_e$ para $e = 1\ldots E$. Por tanto, $E \leq nm$.

Por cada evento, construimos nodos que correspondan con combinaciones admisibles de pedidos, que son procesados en una planta o distribuidos por cualquiera de los vehículos disponibles en el instante correspondiente al evento. Podemos representar cada nodo como un conjunto formado por dos vectores, que representan el estado de las *m* plantas, el primero, y los *V* vehículos, el segundo. La figura 8.5 muestra estos vectores. El vector correspondiente a las plantas indica qué pedidos están siendo procesados. Cada pedido necesita una unidad de capacidad, por tanto el subvector asociado a cada planta *k* tiene C_k posiciones, es decir, C_k pedidos podrían ser procesados al mismo tiempo en esa planta. Por otra parte, el vector asociado con el estado de los vehículos incluye los campos *Origen*, *Destino* y *Pedido*, para representar la planta origen, la planta destino y el pedido que está siendo transportado en el instante asociado al evento. Si un vehículo está inactivo, la planta donde se localiza el vehículo en ese instante se señala en el campo *Origen.*

Plantas														
1				…	*k*				…	*m*				
1	2	…	C_1		1	2	…	C_k		1	2	…	C_m	
				…	*i*	-	…	-	…					

Vehículos								
…	*v*			…	*v'*			…
	Origen	Destino	Pedido		Origen	Destino	Pedido	
…	*k*	*k'*	*i*	…	*k''*	-	-	…

Figura 8.5: Representación de un nodo

Consideremos también un evento de comienzo y otro de fin, cuyos nodos correspondientes son E_s y E_f respectivamente. En estos nodos todas las plantas y vehículos se encuentran sin actividad.

Supongamos que todos los nodos y arcos hasta el evento t_{e-1} han sido ya construidos. Para la construcción del conjunto de nodos para el evento t_e se aplica la siguiente lógica: Supongamos que el evento t_e representa el instante de comienzo del pedido *i* en la planta *k*. Entonces chequeamos todos los nodos en eventos anteriores a t_e , excepto aquéllos que correspondan con el procesamiento del pedido *i* desde otra planta, para evaluar si el subvector asociado a la planta *k* tiene posiciones vacías para poder procesar el pedido *i* y si existe algún vehículo que pudiera estar

disponible para ese instante en la planta *k*. En ese caso, conectamos el nodo a *m* (número de plantas) nuevos nodos que se diferencian en la planta destino del vehículo que distribuye el pedido *i*. El coste para cada uno de esos arcos es igual al beneficio asociado a procesar el pedido *i* desde esa planta, teniendo en cuenta el coste del transporte asociado.

Según la dinámica del grafo y haciendo referencia al cálculo del número de nodos y arcos realizado en escenarios precedentes, el orden en el número máximo de nodos y arcos se determina por el número de pedidos y el tamaño de los vectores de cada nodo. De este modo determinamos el número de nodos y arcos como $O((nm)^{Cm+V})$, C=máximo$\{C_k;\ k=1...m\}$. El chequeo de admisibilidad es equivalente también al de escenarios anteriores, aunque ahora simplemente se buscan posiciones libres en el vector de vehículos y se calcula el mayor grado de simultaneidad en la planta de proceso del pedido.

8.5.2. Un enfoque heurístico

La heurística que ahora presentamos se basa en la idea descrita para el problema en la sección 8.4. El algoritmo comienza resolviendo un problema de flujo a coste mínimo para el grafo definido en esa sección. Sea G^o ese grafo. Una vez obtenida la solución para G^o, chequeamos la admisibilidad de los pedidos seleccionados en cada planta. Para esta tarea, hacemos uso del lema de [Kroon et al, 1995] sobre el problema de la planificación de trabajos fijos (FSP), que nos proporciona una condición necesaria y suficiente para la existencia de un plan admisible para todos los pedidos seleccionados en cada planta:

Sea P_k el conjunto de pedidos seleccionados para ser procesados en la planta *k*. Un plan admisible que incluya a todos los pedidos de P_k existe si y sólo si el solapamiento máximo de los pedidos de P_k es menor o igual a la capacidad C_k de la planta.

Supongamos que los pedidos se procesan en un intervalo de tiempo [1,*T*]. El índice de solapamiento máximo en la planta *k* se define entonces como sigue: L_k = max $\{L_{tk}: 1 \le t \le T\}$ con $L_{tk} = \{i \in P_k: s_{ik} \le t \le l_{ik} - 1\}$. Si L_k excede de C_k entonces la solución proporcionada por el problema de flujo a coste mínimo no es admisible. En este caso, la heurística intentará encontrar un subconjunto de pedidos con beneficio máximo que pueda ser procesado con una capacidad C_k. Este problema es equivalente al problema de maximizar la planificación de trabajos fijos (Max. FSP), descrito en secciones anteriores. El problema se resuelve por un algoritmo de flujo a coste mínimo.

La construcción del grafo G_k que usamos en este trabajo es equivalente a la propuesta para la resolución del problema PDP en el Escenario IB cuando se maximiza el valor de los pedidos, supuesto un número ilimitado de vehículos. Sea el conjunto $R=\{r_e: e = 1...E\}$ usado para representar todos los tiempos de comienzo de los trabajos que pertenecen a P_k en orden cronológico. Es decir, $R= \{s_{ik}: i \in P_k\}$ y $r_{e-1}<r_e$. El conjunto de nodos del grafo se corresponde uno a uno con el conjunto R, además de un nodo final. Existe un arco desde cada nodo al siguiente con coste cero y capacidad ilimitada. Por otra parte, existen arcos desde cada nodo al nodo correspondiente al primer pedido que podría ser producido en la planta, una vez que hubiera finalizado la producción del pedido representado por el nodo que es origen del arco. Estos arcos poseen capacidad para contener una unidad de flujo y el coste es igual a menos el beneficio por producir el pedido en la planta k, es decir, un coste igual a $-(w_i-d_{ik})$. En el primer nodo del grafo se inyectan C_k unidades de flujo, las cuales deben alcanzar el nodo final. Como ejemplo, en la figura 8.7 se muestra el grafo correspondiente al ejemplo de la figura 8.6.

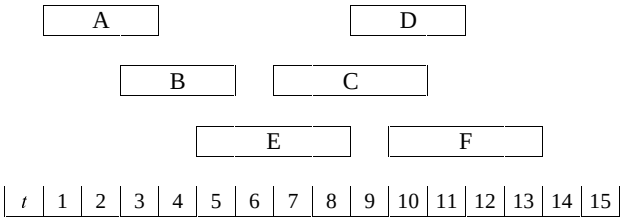


Figura 8.6: Ejemplo de proceso en una planta

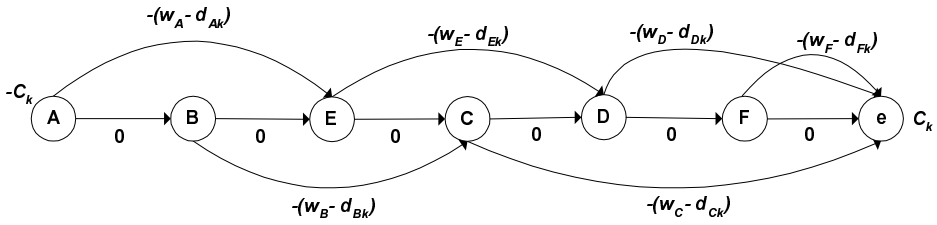


Figura 8.7: Grafo asociado al ejemplo de la figura 8.6

Una vez obtenida la solución óptima de este problema de flujo, denotemos por E_k el conjunto de pedidos de P_k que no han sido seleccionados. Para cada pedido $j \in E_k$, modificamos el grafo G^o para no permitir que el pedido j pueda ser procesado en la planta k. Para tal fin, tenemos que asignar capacidad cero al arco que une n^j_{ok} con n_{js}. Con este nuevo grafo G^o, todo el proceso descrito hasta ahora se repite de nuevo hasta encontrar una solución admisible. El pseudocódigo del algoritmo es el siguiente:

Procedimiento Heurística
  Crear grafo G^o
  Repetir
    Calcular flujo a coste mínimo en G^o
    Para cada planta $k \in K$
      Sea P_k={$i \in P$/ flujo del arco (n^i_{ok} , n_{is}) = 1}, k = 1...m
      Crear grafo G_k
      Calcular flujo coste mínimo en G_k
      Sea E_k = {$j \in P_k$: flujo en el arco (j, posterior(j)) = 0}
      Si $E_k = \varnothing$ entonces
        Admisibilidad en k = *True*
      si no
        Admisibilidad en k = *False*
        Capacidad de los arcos (n^j_{ok},n_{js})=0 en G^o, $j \in E_k$
      Fin Si
    Fin Para
  Hasta que Admisibilidad en k=*True* para $\forall k$
Fin

8.6. RESULTADOS COMPUTACIONALES

En esta sección presentamos y analizamos los resultados obtenidos en los experimentos llevados a cabo.

Para la generación aleatoria de problemas se han considerado los siguientes parámetros:

- Tamaño de los problemas: 20, 30, 40 y 70 pedidos. 10 instancias fueron generadas para cada tamaño.

- Los horizontes de tiempo para el procesamiento de los pedidos se muestran en la Tabla 8.2.

- Los tiempos de producción fueron generados aleatoriamente dentro del rango [1,5]. De la misma forma, los costes de viaje estuvieron dentro del rango [4,10] y los tiempos de descarga dentro del rango [1,3]. El grado de solapamiento por periodo en planta (GSP) y el grado de solapamiento en la fase de distribución (GSD) se muestran en la tabla 8.2. Estas dos medidas permiten la estimación del grado de “estrechez” de los pedidos. En el Anexo I viene definido el procedimiento de cálculo de estos parámetros.

- Los valores de los pedidos se generaron aleatoriamente dentro del rango [30,100]

n	Horizonte temporal	Promedio PSP	Promedio PSD
20	[1,70]	0.80	3.62
30	[1,80]	0.93	4.25
40	[1,90]	1.02	5.36
70	[1,120]	1.23	6.02

Tabla 8.2: Horizonte temporal y promedios de solapamiento

- Para los experimentos llevados a cabo se tomaron varios valores para el número de vehículos y el número de plantas. La tabla 8.3 presenta estos datos.

n	*m*	*V*
20	2, 3	2, 3, 4
30	2, 3	2, 3
40	2, 3	2, 3
70	2	2

Tabla 8.3: Número de vehículos y plantas en función de *n*

- En todas las ejecuciones se supuso que cualquier pedido podría ser procesado en cualquiera de las plantas, es decir, $K_i = K \ \forall i$. El número de vehículos iniciales en cada planta se obtuvo de forma aleatoria.

- La capacidad de producción en las plantas se consideró 1 para todos los problemas.

- Todos los tiempos vienen dados en segundos de CPU sobre un Pentium III a 850 Mhz.

- Los porcentajes de error se computan respecto al valor de la solución óptima mediante la siguiente fórmula:

$$\frac{\text{(Solucion optima - Solucion heuristica)}}{\text{Solucion optima}}$$

8.6.1. Comportamiento del método exacto

Para este escenario, el uso del método de resolución exacto se muestra bastante ineficiente. Para la batería de problemas generados, no es posible la resolución con más de 3 plantas, incluso suponiendo capacidad C=1 en todas ellas. Por ello, la razón fundamental del desarrollo de este método es comparar los resultados obtenidos con el procedimiento heurístico, y de esa forma predecir su comportamiento con problemas mayores.

La tabla 8.4 muestra el número de nodos, número de arcos y los tiempos de computación para una instancia de cada tamaño resuelto.

n	*m*	*V*	Nº nodos	Nº arcos	Tiempo
20	2	2	3290	21944	16
20	3	2	12718	74210	69
20	2	3	28898	198060	378
20	2	4	236052	1216054	19640
30	2	2	5736	64229	44
30	3	2	41726	659519	679
30	2	3	245791	2463023	30300
40	2	2	14799	233562	208
40	3	2	75870	1931667	3185
40	2	3	287277	3578476	38700
70	2	2	84232	3816991	6531

Tabla 8.4: Comportamiento del método exacto

Como puede verse en la tabla, problemas con más de 3 vehículos requieren además de un gran consumo de memoria, un excesivo tiempo de computación únicamente con 20 pedidos. Problemas con más de 5 vehículos y 30 pedidos requerirían más de un día para su resolución.

8.6.2. Comportamiento del procedimiento heurístico

Con respecto a la heurística descrita, la tabla 8.5 presenta los resultados de las ejecuciones realizadas. En todos los casos, los datos son valores promedio respecto a las 10 instancias de cada tamaño de problema.

n	*m*	*V*	Error (%)	Nº éxitos	Tiempo	Nº iteraciones
20	2	2	0.39	7	0.9	0.9
20	3	2	1.03	7	1	1
20	2	3	1.60	5	1.9	1.4
20	2	4	2.18	2	2.4	3.6
30	2	2	0.86	6	1	0.8
30	3	2	0.41	8	0.9	0.5
30	2	3	1.31	3	2	2.5
40	2	2	1.25	5	1.1	0.9
40	3	2	0.87	7	1.3	0.7
40	2	3	2.04	3	3.6	3
70	2	2	1.37	5	2.1	1.6

Tabla 8.5: Sumario de resultados

A la vista de los resultados se pueden extraer las siguientes conclusiones:

- El error obtenido por la heurística presenta un crecimiento mínimo respecto al aumento en el número de recursos. Las peores instancias fueron para V=4 y n=20, aunque el error siempre fue menor del 2.2%.

- Respecto a las iteraciones realizadas por el método para obtener la solución, los resultados muestran que el error se incrementa con el incremento del número de iteraciones llevadas a cabo, lo cual es lógico, puesto que un mayor número de iteraciones significa mayor dificultad para obtener admisibilidad en todas las plantas.

- Respecto a la habilidad para obtener soluciones óptimas, 58 de las 110 instancias fueron resueltas con éxito.

- Los tiempos de computación de la heurística son bastante bajos, nunca superando los 4 segundos.

Por todo lo anterior, podemos concluir que la heurística asegura buenos resultados en un breve tiempo de computación, y por ello, podría ser usado con cierta garantía en problemas de mayor tamaño y mayor número de recursos.

Capítulo 9

Escenario IV: El problema PDP con ventanas temporales de entrega y varias plantas de producción

Índice

9.1. INTRODUCCIÓN

El Escenario IV se ocupa del problema PDP con varias centros de producción y ventanas temporales de entrega. Sería el escenario general del problema, donde los escenarios anteriores podrían ser considerados como casos particulares del mismo. En síntesis, este escenario se ocupa del problema de seleccionar y planificar un conjunto de pedidos para ser fabricados e inmediatamente distribuidos, haciendo uso de varias plantas de producción y asumiendo plazos de entrega variable para los pedidos solicitados.

El contenido del capítulo es el siguiente:

- En la sección 9.2 se plantean brevemente las características del problema PDP en este escenario

- En la sección 9.3 se describe la formulación del problema, atendiendo a la formulación expresada en el Escenario III.

- En la sección 9.4 se presenta, a partir de los métodos descritos en el escenario II y III del problema, un método de solución exacta basado en grafos. Este método exacto es utilizado para obtener las soluciones óptimas de los problemas y medir el comportamiento del algoritmo genético descrito en la siguiente sección. El método exacto puede únicamente ser usado con instancias pequeñas.

- En la sección 9.5 se describe un algoritmo genético que contiene un enfoque heurístico para el cálculo del fenotipo de los individuos. Dicho algoritmo se diseña a partir del algoritmo genético II descrito en el Escenario II del problema y del enfoque heurístico planteado en el Escenario III.

- En la sección 9.6 se presentan los resultados computacionales sobre una batería de problemas generada aleatoriamente a partir del concepto de simultaneidad descrito en el Anexo I.

9.2. DESCRIPCIÓN DEL PROBLEMA

A continuación apuntamos brevemente las características y datos asociados al problema en el escenario IV. Las características en este escenario se corresponden con las ya apuntadas en el Escenario III del problema, agregando la ventana temporal

$[A_i, B_i]$ para la entrega de cada pedido.

El problema considera la selección y planificación de un conjunto dado de pedidos. Los pedidos seleccionados deben ser preparados en una cualquiera de las plantas de producción existentes, e inmediatamente distribuidos al lugar del cliente. La entrega del pedido debe producirse dentro de una ventana temporal, existiendo un fecha ideal de entrega dentro de dicha ventana. Para la producción existe un conjunto de m plantas $K=\{1...m\}$, cada una con una capacidad de producción limitada C_k. Si la planta k tiene C_k unidades de capacidad, entonces a lo sumo C_k pedidos pueden ser preparados simultáneamente en dicha planta. Además, cada pedido i puede únicamente ser preparado en un subconjunto K_i del conjunto de plantas K.

Para la distribución se cuenta con un número fijo V de vehículos de idénticas características, considerando v_k como el número de vehículos inicialmente en la planta k. Por tanto, $\sum_{k=1}^{m} v_k = V$

El número de pedidos, sus tiempos de proceso en planta y los tiempos de viaje desde las localizaciones de destino a cualquiera de las plantas son datos conocidos y fijos. De ese modo, con cada pedido i, del conjunto de pedidos $P = \{1...n\}$, se asocian los siguientes datos:

- Una ventana temporal de entrega $[A_i, B_i]$.
- Una fecha ideal de entrega e_i dentro de la ventana $[A_i, B_i]$.
- Un tiempo ti_{ik} desde cada planta k a la localización del pedido.
- Un tiempo de viaje tr_{ik} desde la localización hasta cada planta.
- Un tiempo de proceso o producción tp_i independiente de la planta.
- Un tiempo de descarga tu_i.

Puesto que todos los tiempos de proceso (tiempos de producción y distribución) son conocidos de antemano, cada ventana temporal de entrega $[A_i, B_i]$ puede ser trasladada al comienzo del procesamiento en cada planta, de forma que contemos con una ventana temporal de comienzo $[a_{ik}, b_{ik}]$ en cada planta. Podemos lógicamente trasladar también la fecha ideal de entrega a un instante ideal de comienzo s_{ik} en cada planta. La Figura 9.1 muestra el proceso gráfico asociado a la actividad de un pedido servido desde la planta k cuando el vehículo retorna a la

planta *k'*.

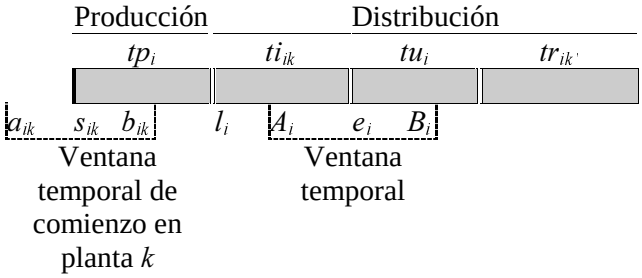


Figura 9.1: Actividad de un pedido

El objetivo en el problema es la maximización del beneficio asociado a la selección de los pedidos, asumiendo que cuando un pedido no es servido en su instante ideal de entrega, se produce un decremento del valor del pedido proporcional a la desviación. Sea W_i el beneficio asociado con servir el pedido i en su instante ideal e_i y sean w_i^- y w_i^+ penaltis por entrega anticipada y retrasada del pedido, respectivamente. Estos penaltis son usados para decrementar el valor del pedido cuando es servido antes o después de su instante ideal. De ese modo, cuando un pedido se sirve en el instante t, el beneficio en ese instante viene a ser el siguiente:

$$w_i = \begin{cases} W_i - (t - e_i)w_i^+ & \text{si } t \geq e_i \\ W_i - (e_i - t)w_i^- & \text{si } t < e_i \end{cases}$$

9.3. FORMULACIÓN DEL PROBLEMA

Para la formulación del problema, y al igual que ocurría en los escenarios II y III del problema, se introduce una escala de tiempo discreto dividiendo el tiempo en T periodos, con $t = 1...T$. Para cada pedido $i \in P$ y sobre cada planta $k \in K_i$ se consideran los posibles instantes de comienzo $r_{ijk} \in [a_{ik}, b_{ik}]$ definidos por un índice de comienzo j que recorre el número de posibles instantes de comienzo del pedido, $j = 1...(b_i - a_i + 1)$.

Un dato necesario en la formulación y calculado en función del instante de comienzo y de los tiempos de viaje es f_{ijk}, el cual define el instante de finalización de la actividad de un pedido i con instante de comienzo de índice j, cuando el vehículo con el que fue servido retorna a la planta k. De forma análoga denotamos l_{ijk} como el instante de comienzo de la fase de distribución del pedido con instante de comienzo definido por el índice j.

Se define S(t,k)= {(i,j): $i \in P$ $j \in [1... (b_i-a_i+1)]$/ $r_{jjk} \leq t$ & $r_{jjk} + tp_i > t$} como el conjunto de pares (*pedido*, *índice de instante de comienzo*) que definen los pedidos, según instante de comienzo, cuyas fases de producción estarían solapadas en el instante t del horizonte de planificación sobre la planta k. Se define un conjunto para cada $t \in [1...T]$ y para cada $k \in K$.

Por otra parte, sea d_{ik} el coste de transportar el pedido i desde la planta k, $d_{ik} = ti_{ik}$, y d'_{ik} el coste de la vuelta del vehículo a la planta desde la localización del pedido i, $d'_{ik} = tr_{ik}$.

- **Variables**

Nuestro modelo incorpora las siguientes variables de decisión:

- x_{ijk} = 1 si el pedido i se procesa en la planta k en el instante de índice j; 0 en otro caso. ($\forall i \in P, j \in [1... (b_i-a_i+1)], \forall k \in K_i$)

- z_{ijk} = 1 si el vehículo que sirve el pedido i con instante de comienzo de índice j regresa a la planta k; 0 en otro caso. ($\forall i \in P$; $j \in [1... (b_i-a_i+1)]$; $\forall k \in K$)

- y_{tk} = número de vehículos disponibles en el instante t en la planta k , $t \in \{1..T\}$.

- **Restricciones**

1. Cada pedido puede ser procesado a lo sumo en una planta y con un único instante de comienzo:

$$\sum_{k \in K_i} \sum_{j=1}^{b_i - a_i + 1} x_{ijk} \leq 1 \quad \forall i \in P$$

2. Tras la descarga de un pedido una vez servido, el vehículo debe retornar a una cualquiera de las plantas:

$$\sum_{k \in K_i} x_{ijk} - \sum_{k=1}^{K} z_{ijk} = 0 \quad \forall i \in P \quad \forall j \in [1, b_i - a_i + 1]$$

3. Número de vehículos iniciales en cada planta

$$y_{0k} = v_k \quad \forall k \in K$$

4. Cómputo del número de vehículos disponibles en cada planta para cada instante *t*.

$$y_{tk} = y_{0k} - \sum_{i,j/r_{ijk}+tp_i<t} x_{ijk} + \sum_{i,j/f_{ijk}\le t} z_{ijk} \qquad \forall t \in \{1..T\}; \forall k \in K$$

5. Deben existir vehículos disponibles en el instante l_{ijk} para poder procesar el pedido *i* en la planta *k*.

$$\sum_{(i,j)/l_{ijk}=t} x_{ijk} \le y_{tk} \qquad \forall t \in \{1..T\}; \forall k \in K$$

6. No más de C_k pedidos pueden ser asignados al mismo tiempo, en cualquier instante, en la planta *k*.

$$\sum_{(i,j)\in S(t,k)} x_{ijk} \le C_k \quad \forall k \in K_i; \forall t \in [1...T]$$

7. Cálculo de los instantes de adelanto o retraso de los pedidos.

$$\sum_{k\in K_i} \sum_{j=1}^{b_i-a_i+1} (s_{ik} - b_{ik} - j + 1) x_{ijk} + n_i^- - n_i^+ = 0 \qquad i = 1\ldots n$$

El modelo completo para una función objetivo que maximiza el beneficio de la planificación es el siguiente:

$$Max \sum_{i=1}^{n} \sum_{k \in K_i} w_i x_{ik} - \sum_{i=1}^{n} \left(w_i^+ n_i^+ + w_i^- n_i^- \right) - \left(\sum_{i=1}^{n} \sum_{k \in K_i} d_{ik} x_{ik} + \sum_{i=1}^{n} \sum_{k=1}^{K} d'_{ik} z_{ik} \right)$$

sujeto a

$$\sum_{k \in K_i} \sum_{j=1}^{b_i - a_i + 1} x_{ijk} \leq 1 \quad \forall i \in P$$

$$\sum_{k \in K_i} x_{ijk} - \sum_{k=1}^{K} z_{ijk} = 0 \quad \forall i \in P \quad \forall j \in [1, b_i - a_i + 1]$$

$$y_{0k} = v_k \quad \forall k \in K$$

$$y_{tk} = y_{0k} - \sum_{i,j / r_{ijk} + tp_i < t} x_{ijk} + \sum_{i,j / f_{ijk} \leq t} z_{ijk} \qquad \forall t \in \{1..T\}; \forall k \in K$$

$$\sum_{(i,j)/l_{ijk}=t} x_{ijk} \leq y_{tk} \qquad \forall t \in \{1..T\}; \forall k \in K$$

$$\sum_{(i,j) \in S(t,k)} x_{ijk} \leq C_k \quad \forall k \in K_i; \forall t \in [1...T]$$

$$\sum_{k \in K_i} \sum_{j=1}^{b_i - a_i + 1} (s_{ik} - b_{ik} - j + 1) x_{ijk} + n_i^- - n_i^+ = 0 \qquad i = 1 \ldots n$$

$x_{ijk}, z_{ijk} \in \{0,1\}$ y_{tk} Enteras

9.4. MÉTODO EXACTO DE RESOLUCIÓN BASADO EN GRAFOS

El enfoque que presentamos está basado en los métodos exactos descritos para los escenarios II y III del problema. El método construye un grafo *G* que colecciona las soluciones admisibles del problema a través de un método de evaluación de estados admisibles en la secuenciación de los pedidos. El camino máximo desde el nodo de inicio al nodo final en *G* proporciona la solución óptima del problema.

El proceso puede ser descrito como sigue: Consideremos los eventos correspondientes a los instantes de inicio de cada pedido en cada planta. Sea *D* el número total de esos eventos. El conjunto $\{t_d / d = 1,\ldots, D\}$ es usado para representar esos eventos en orden cronológico. Es decir, $\{t_d / d = 1,\ldots, D\} = \{r_{ijk} / r_{ijk} \in [a_{ik}, b_{ik}]; j = 1... (b_i - a_i + 1), i = 1 \ldots n, k = 1 \ldots m\}$ y $t_{d-1} \leq t_d$ para $d = 1 \ldots D$. Por tanto, el número total de eventos es $D = nm \sum_{i=1}^{n} (b_i - a_i + 1)$.

Para cada evento se crean nodos correspondientes a combinaciones legales de pedidos que están siendo procesados en una planta o distribuidos sobre uno

cualquiera de los vehículos disponibles en ese instante en la planta correspondiente al evento. Cada nodo contiene dos vectores representando el estatus de las *m* plantas y de los *V* vehículos (Figura 9.2). El vector correspondiente a las plantas indica qué pedidos están siendo procesados en el instante correspondiente al evento. Cada pedido necesita una unidad de capacidad, por tanto el subvector asociado a cada planta *k* contiene C_k posiciones, es decir, C_k pedidos podrían ser procesados en un mismo instante en esa planta. Por otra parte, el vector asociado con los vehículos incluye la planta donde se localiza cada vehículo en el instante del evento (*Origen*) y, para los vehículos que están distribuyendo algún pedido, la información asociada con el envío (el evento correspondiente r_{ijk} que define el pedido, el instante de inicio y la planta, además de la planta destino donde finalizará el vehículo después de servir el pedido).

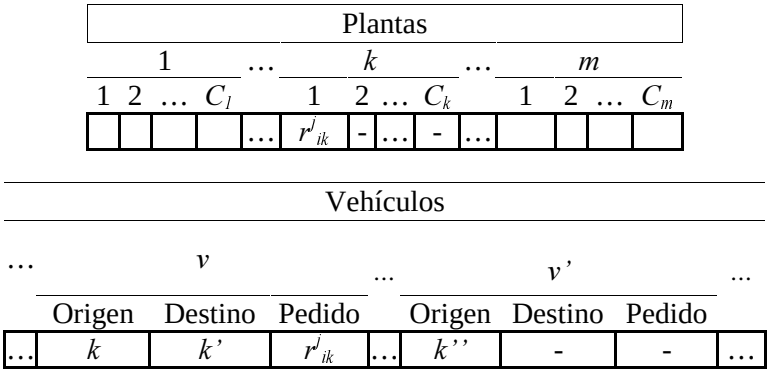


Figura 9.2: Representación de un nodo

En el grafo se considera un evento de inicio y finalización representados respectivamente por los nodos E_s y E_f. Estos nodos no poseen actividad alguna.

Supongamos que todos los nodos hasta el evento $t_{d\text{-}1}$ han sido ya construidos. Sea el evento t_d el correspondiente al instante número *j* de inicio del pedido *i* en la planta *k*. Para construir el siguiente conjunto de nodos para el evento t_d se aplica la siguiente lógica: Para cada nodo *n* correspondiente a cada uno de los eventos anteriores, t_s s = 1...d-1, con t_s un evento correspondiente a un pedido $j \neq i$, se actualiza el estado de las plantas y de los vehículos en el instante correspondiente al evento t_d y se evalúa si el subvector asociado a la planta *k* tiene posiciones libres para procesar el pedido *i*, y si existe algún vehículo libre localizado en la planta *k*. En caso afirmativo, se conecta el nodo *n* a *m* nuevos nodos que difieren en la planta de destino del vehículo que distribuye el pedido. Los nodos se conectan con arcos de peso igual al beneficio asociado a procesar el pedido *i* en el instante número *j* en la planta *k*.

El número de nodos y arcos del grafo, atendiendo al cálculo realizado en escenarios anteriores, en concreto al cálculo realizado en los escenarios II y III del problema, resulta ser de orden $O((nmk)^{Cm+V})$ con k = máximo$\{k_i: k_i = b_i - a_i + 1\}$.

9.5. ALGORITMO GENÉTICO PROPUESTO

A partir del algoritmo genético implementado para la resolución del Escenario II del problema y junto con la heurística definida en el Escenario III, presentamos un enfoque basado en algoritmos genéticos para la resolución del Escenario IV. El buen comportamiento obtenido con los métodos anteriormente señalados lleva a la implementación de este enfoque heurístico. A continuación se describen los elementos del algoritmo:

- **Representación de los individuos**

Un *string* será usado para representar el genotipo de los individuos. La longitud del *string* es 2*n*, con *n* el número de pedidos solicitados. Cada par de posiciones contiguas en el *string*, denotadas como gen, corresponden a cada pedido y contiene, en primer lugar, el instante en el que el pedido debería comenzar, y en segundo lugar la planta donde sería procesado. Este instante se define con respecto al instante ideal de entrega del pedido. La figura 9.3 muestra el genotipo de un individuo para un problema con 5 pedidos y 2 plantas. Para ese individuo, el pedido 2 debería iniciarse en el instante $r_{2j1} = s_{21}-1$, el pedido 3 en el instante $r_{3j2} = s_{32}+1$ y el resto de pedidos en sus instantes ideales de inicio. Los pedidos 1, 2 y 5 serían procesados en la planta 1 y el resto en la planta 2.

pedido 1		pedido 2		pedido 3		pedido 4		pedido 5	
0	1	-1	1	1	2	0	2	0	1

Figura 9.3. Ejemplo de genotipo

Debido a la limitación de recursos (capacidad de producción y número de vehículos), no tenemos garantía de que todos los pedidos puedan ser servidos usando esos instantes de inicio y esas plantas. Por ello, se busca un subconjunto de pedidos que maximice el beneficio total. Esta situación representa un caso particular del problema de la producción y distribución conjunta con instantes fijos de entrega y varias plantas, es decir, la situación planteada en el Escenario III del problema. Para la obtención del fenotipo del individuo aplicamos una variante del método heurístico propuesto para la resolución del problema en el Escenario III. El hecho de usar una variante radica en que la situación que planteamos aquí asigna ya una planta para la producción del pedido, punto éste no considerado en dicha heurística.

La variante se basa en la observación de que el problema puede ser modelado como un problema de flujo a coste mínimo si la capacidad de producción es suficiente para procesar todos los pedidos que pueden ser distribuidos, observación definida en la sección 8.4. Por ello, la heurística parte de la idea de no considerar las restricciones asociadas a la capacidad de producción en las plantas.

El algoritmo comienza con la formación de un grafo que representa la asignación establecida en el genotipo del individuo. Para ese grafo, la solución de flujo a coste mínimo proporciona la asignación óptima de vehículos que maximiza la planificación de esos pedidos, sin tener en cuenta la producción previa de los mismos. La construcción del grafo directo G usado para resolver esta primera parte de la heurística se describe como sigue: Para cada planta, existe un nodo en G. Representamos esos nodos como $\{n_1 \ldots n_k \ldots n_m\}$. Para cada pedido i, creamos un subgrafo G_i con la siguiente estructura: Nodos n_{is} y n_{ie} definen el inicio del procesamiento en planta definida en el genotipo y el fin de la descarga, respectivamente. El valor del pedido i se representa como un coste negativo $-c_i$ en el arco que conecta n_{is} y n_{ie}. El valor c_i se calcula con respecto al instante de comienzo r_{ijk} definido en el genotipo, mediante la siguiente expresión:

$$c_i = w_i - d_{ik} = \begin{cases} W_i - (r_{ijk} - e_i)w_i^+ - d_{ik} & \text{si } r_{ijk} \geq s_{ik} \\ W_i - (e_i - r_{ijk})w_i^- - d_{ik} & \text{si } r_{ijk} < s_{ik} \end{cases}$$

Nodos n^i_{rk}, $k = 1 \ldots m$, representan la planta de retorno para el vehículo que sirve el pedido i. Cada nodo n_{ie} se conecta a cada nodo n^i_{rk}, $k = 1 \ldots m$, con capacidad uno y coste igual a menos el coste de volver a la planta k desde la localización del pedido i, es decir, $-d'_{ik}$. Puesto que un pedido es procesado a lo sumo una vez, la capacidad para cada arco en G_i es uno. Para cada nodo de planta n_k, existen arcos, con una unidad de capacidad y coste cero, que conectan todos los nodos n_{is} para los pedidos i que se han asignado a la planta k. Además, existe un arco desde cada nodo n^i_{rk} a todo nodo $n_{i'k}$ siempre que el instante de comienzo del pedido i' sea mayor que el instante en el que el vehículo retorna a la planta k después de servir el pedido i. Estos arcos tienen también coste nulo y pueden transportar una unidad de flujo. Todos los nodos n^i_{rk}, $i = 1 \ldots n$, $k = 1 \ldots m$, se conectan a un nodo final e con coste cero y una unidad de capacidad.

Si inyectamos v_k (es decir, el número de vehículos inicialmente en la planta k) unidades de flujo en el nodo n_k, $k = 1 \ldots m$, la solución óptima al problema de flujo a coste mínimo en G devolverá la secuenciación de un subconjunto de pedidos de valor total máximo. Un pedido i es servido si y sólo si en la solución del problema de flujo, una unidad de flujo pasa a través del arco (n_{is}, n_{ie}).

A continuación se presenta una ilustración del método de construcción. En la tabla 9.1 se presentan los datos correspondientes a 5 pedidos solicitados.

Pedido	d_{ik}	$d_{ik'}$	W_i	w_i^-	w_i^+	w_i	c_i	r_{ijk}	f_{ij1}	f_{ij2}
1	5	5	17	1	1	17	12	3	11	13
2	5	5	25	1	1	24	19	4	17	16
3	5	5	15	1	1	14	9	4	13	15
4	5	5	18	1	1	13	13	13	22	21
5	5	5	15	1	1	10	10	15	21	18

Tabla 9.1: Ejemplo para construcción del grafo *G*

La figura 9.4 representa el grafo construido en la primera parte del método heurístico, el cual corresponde al genotipo representado en la figura 9.3. Se cuenta con 2 vehículos para el transporte, inicialmente uno en cada planta. Los datos f_{ijk} representan los instantes en los cuales los vehículos retornarían a la planta k tras servir el pedido i. Los datos r_{ijk} representan el instante de comienzo según el instante y la planta definidos en el genotipo. Las líneas gruesas en *G* denotan el camino de flujo a coste mínimo. Los pedidos seleccionados son el 1, 2, 4 y 5.

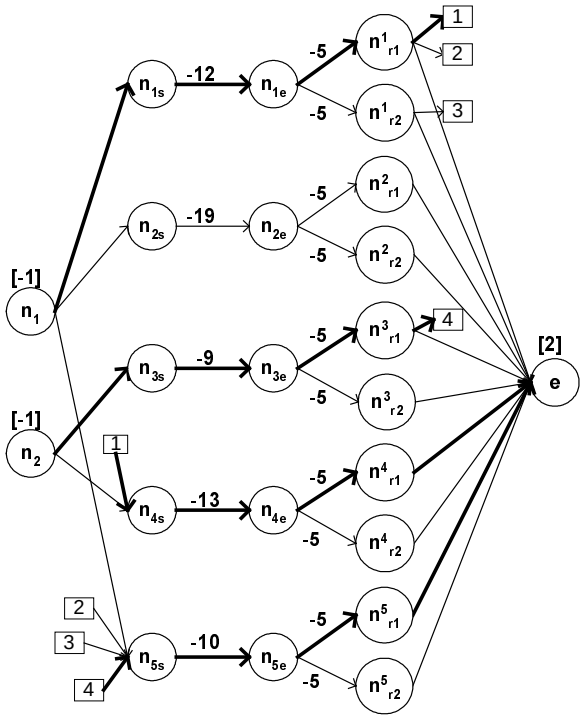


Figura 9.4: Grafo *G* correspondiente al genotipo de Figura 9.3

Una vez que se obtiene una solución para *G*, se chequea admisibilidad en cada una de las plantas de la misma forma apuntada ya en el método heurístico del escenario III. Describimos brevemente este proceso:

Sea P_k el conjunto de pedidos seleccionados para ser procesados en la planta k. Un plan admisible que incluya todos los pedidos pertenecientes a P_k existe si y sólo si el

máximo grado de simultaneidad de P_k es menor o igual a la capacidad C_k en la planta. El máximo grado de simultaneidad de los pedidos se define como sigue: $L = \max \{L_t: 0 \leq t \leq T\}$ con $L_t = \{i \in P_k: r_{ijk} \leq t \leq r_{ijk} + tp_i\}$, $r_{ijk} \in [a_{ik}, b_{ik}]$.

Si el máximo índice de solapamiento de P_k excede de C_k entonces la solución proporcionada por el problema de flujo a coste mínimo no es admisible. En este caso, la heurística intentará encontrar un subconjunto de pedidos con beneficio máximo que pueda ser procesado con una capacidad de C_k, lo cual corresponde a la resolución de un problema FSP en cada planta, mediante un algoritmo de flujo a coste mínimo. Una vez obtenida la solución a este problema de flujo sobre el grafo G_k construido previamente con la misma filosofía descrita en escenarios anteriores, sea E_k el conjunto de los pedidos de P_k que no han sido seleccionados. Para cada pedido $i \in E_k$, se modifica el grafo inicial G, con objeto de prohibir que el pedido i pueda ser procesado en la planta k. Para este fin, simplemente se necesita asignar capacidad cero al arco que une n_{is} con n_{ie}. Con este nuevo grafo G, se repite todo el proceso descrito hasta ahora hasta encontrar una solución admisible tanto en asignación de vehículos como en consumo de capacidad en cada planta.

El pseudocódigo del procedimiento heurístico para la obtención del fenotipo sería el siguiente:

```
Crear Grafo G
Repetir
        Calcular flujo a coste mínimo en G
        Para cada planta k
                Sea Pk={i/ flujo en arco  nis→nie = 1}
                Crear grafo Gk
                Calcular flujo a coste mínimo en Gk
                Ek = {j∈Pk / flujo en arco (j, posterior(j)=0}
                Si Ek = ∅ entonces
                    Admisibilidad en k = true
                  si no
                    Admisibilidad en  k = false
                    Capacidad de arcos (nis→nie) = 0 in G, j∈Ek
                Fin Si
Hasta Admisibilidad en k = true, ∀k
```

- **Selección y reproducción de la población**

En nuestro algoritmo hemos usado la variante *steady-state*, reemplazando un individuo en cada iteración. Por tanto, en cada iteración se genera un nuevo individuo usando los operadores descritos en la siguiente sección. En cada iteración un operador será seleccionado con una cierta probabilidad de ser elegido.

Para seleccionar el individuo a ser reemplazado, usamos un enfoque basado en un ranking exponencial de los individuos de la población. El ranking exponencial asigna al peor individuo (peor *fitness*) una probabilidad de ser eliminado igual a *p*. Si éste no es seleccionado, entonces se asigna la misma probabilidad al siguiente con peor *fitness*, y así sucesivamente.

- **Operadores de cruce y mutación**

El operador de cruce es el operador de combinación más importante para generar nuevos individuos, es decir, nuevos puntos de búsqueda. Se seleccionan dos individuos llamados padres y se produce, intercambiando partes de los padres, un nuevo individuo llamado *offspring* o hijo. Dados dos individuos padres *p1* y *p2* seleccionados aleatoriamente, el hijo se ha obtenido aplicando la siguiente regla: Sea $p1(j)$ el gen o posición j en *p1*. Para cada j desde 1 a $2n$ (número de posiciones), si $p1(j) = p2(j)$ entonces $hijo(j) = p1(j)$, en otro caso $hijo(j)$ es un valor aleatorio en el intervalo $[p1(j), p2(j)]$. En la figura 7 se puede ver un ejemplo del operador de cruce.

p1	-1	2	0	1	0	1
p2	-1	1	1	1	0	3
Hijo	-1	Aleat[1,2]	Aleat [0,1]	1	0	Aleat [1,3]

Figura 9.5: Operador de cruce

También se ha empleado un operador estándar de mutación que selecciona aleatoriamente un individuo y, aleatoriamente también, elige un nuevo valor para una de sus posiciones.

9.6. RESULTADOS COMPUTACIONALES

En esta sección presentamos y analizamos los resultados obtenidos en los experimentos llevados a cabo. La primera etapa en los experimentos involucró la generación aleatoria de una batería de problemas (sección 9.6.1). En la sección 9.6.2 se muestran 4 instancias características del comportamiento del método exacto. En la sección 9.6.3 se definen los parámetros del algoritmo y se presentan los resultados obtenidos respecto a las soluciones óptimas proporcionadas por el método de solución exacta.

9.6.1. Generación de problemas

Para el estudio de los métodos de resolución descritos se generó una batería de problemas con las siguientes características:

- Tamaños y número de problemas: Se consideraron cuatro tamaños de problemas, n = 10, 20, 30 y 40 pedidos. Diez instancias fueron aleatoriamente generadas para cada tamaño de problema.

- Ventanas Temporales: Los valores para $[a_i,b_i]$ fueron generadas aleatoriamente con tamaños entre 1 y 5 instantes de tiempo.

- El horizonte temporal de los problemas se consideró dependiente del número de pedidos, de acuerdo a los siguientes intervalos:

n	Intervalo
10	[1,65]
20	[1,80]
30	[1,90]
40	[1,100]

Tabla 9.2: Horizonte temporal de los pedidos

Ello dio lugar a los siguientes promedios de simultaneidad. El cálculo del promedio de simultaneidad en cada escenario viene descrito en el Anexo I del documento.

n	L^P_m	L^D_m
10	0.53	2.18
20	0.76	3.58
30	0.97	4.83
40	1.08	5.52

Tabla 9.3: Promedios de simultaneidad de los pedidos

- Los valores W_i fueron también generados aleatoriamente dentro del intervalo [30,100].

- Los penaltis por retraso w_i^+ y adelanto w_i^- fueron seleccionados dentro del intervalo [0,2].

- La capacidad de producción C_k para cada planta en todos los problemas fue 1. Se consideraron 2 plantas de fabricación y 2 vehículos.

- El número de vehículos inicialmente en cada planta fue generado también aleatoriamente.

- Todos los tiempos vienen dados en segundos de CPU sobre un Pentium III a 850 Mhz.

- Los porcentajes de error se computan respecto al valor de la solución óptima mediante la siguiente fórmula:

$$\frac{\text{(Solucion optima - Solucion heuristica)}}{\text{Solucion optima}}$$

9.6.2. Comportamiento del método exacto

La tabla 9.4 muestra, para cuatro instancias características, el comportamiento del método exacto con relación al número de nodos y arcos y al tiempo de computación.

Instancia	*n*	Nº de nodos	Nº de arcos	Tiempo
1	10	6764	41780	22
2	20	37937	561692	1081
3	30	64179	1347160	2530
4	40	140781	6814809	16740

Tabla 9.4. Comportamiento del método exacto

Puede claramente apreciarse el crecimiento del grafo con respecto tanto al número de pedidos. Instancias mayores o con mayor de número de vehículos o plantas no fueron consideradas debido al aumento exponencial del tamaño del grafo.

9.6.3. Resultados del algoritmo genético

Se usaron los siguientes parámetros para el algoritmo:

- Población inicial obtenida de un modo aleatorio

- Tamaño de la población = 20

- Probabilidad de cruce = 0.6

- Probabilidad p en el ranking exponencial = 0.2

- Número de iteraciones = 1000

La Tabla 9.5 muestra el sumario de resultados obtenidos por el algoritmo genético. En todos los datos se ha tomado promedio sobre las 10 instancias resueltas de cada tamaño de problema *n*.

n	Nº medio de pedidos servidos	Error medio (%)	Nº soluciones óptimas	Tiempo medio de computación
10	5.1	0.74	9	398
20	6.9	1.09	7	502
30	8.2	0.59	5	710
40	9	2.12	4	923

Tabla 9.5: Resultados del algoritmo genético

Comentarios

De la resolución de los problemas se extraen las siguientes conclusiones:

- El algoritmo genético obtuvo éxito en 25 de los 40 problemas resueltos. El promedio de error nunca excedió del 2.2%.

- Respecto a los tiempos de computación, el método exacto empleó un tiempo excesivamente mayor al del algoritmo genético, a pesar de que la heurística implementada es la que presenta peores tiempos de entre todas las descritas en la tesis.

Capítulo 10

Conclusiones y extensiones

Índice

10.1. CONCLUSIONES Y APORTACIONES

En esta investigación se ha presentado un problema de planificación de pedidos con producción y distribución inmediata. El problema, al cual se ha denominado Planificación conjunta de la producción y distribución de pedidos (PDP), considera la problemática asociada con productos no almacenables en un sistema con recursos limitados de producción y distribución. A pesar de ser relevante la problemática asociada a la fabricación y distribución de productos no almacenables, el problema no ha sido lo suficientemente estudiado en la literatura. El enfoque práctico del problema se centra en la problemática asociada a la distribución de hormigón *ready-mix*.

El problema ha sido considerado como un problema de secuenciación de trabajos en máquinas. Los modelos de secuenciación que responden a la situación planteada en esta investigación han sido los siguientes:

- Secuenciación de trabajos con intervalos de procesamiento: Los problemas contemplados han sido el problema FSP (programación de trabajos fijos) y el problema VSP (programación de trabajos variables). El problema FSP ha tenido mucha relevancia en esta investigación, pues se ajusta completamente a las características del problema PDP en uno de los escenarios en los que se ha planteado.

- Sistemas de flujo uniforme con máquinas en paralelo (FSPM) con *no-wait* en proceso en entornos *just-in-time*: De la revisión bibliográfica realizada en torno a la filosofía *just-in-time* dentro de la secuenciación de trabajos, se ha detectado que la mayoría de los trabajos se presentan en entornos con una sola máquina o máquinas en paralelo. Sin embargo, no se ha considerado esta filosofía en los sistemas FSPM.

El problema ha recibido diversos enfoques que se estructuran sobre cuatro escenarios principales. A continuación se muestran las aportaciones realizadas en cada escenario, estructuradas por secciones:

10.1.1. Escenario I

El Escenario I presenta el problema PDP con instantes fijos de entrega y una planta de producción. El objetivo del problema es la maximización del beneficio. Un análisis del problema en este escenario desglosa el estudio en dos escenarios distintos:

- **Escenario IA**

Se estudia el problema PDP con un número fijo de vehículos. El problema se traduce en este escenario en un problema FSPM con objetivo la maximización del número de trabajos finalizados en su fecha de entrega, problema nunca antes considerado.

Se presenta un modelo de programación entera y un método de solución exacta basado en la construcción de un grafo que recoge las soluciones admisibles del problema. Sobre el grafo se aplica una regla de reducción de arcos para eliminar soluciones admisibles que las características del problema descartan como posibles soluciones óptimas del mismo. El camino más largo en el grafo provee la solución óptima del problema. Las características más importantes de la construcción del grafo son:

- El proceso de construcción garantiza la formación de un grafo enumerado, lo cual contribuye a la rapidez en el cálculo del camino máximo.

- La regla de reducción del grafo reduce notablemente la dimensión del mismo. Los experimentos realizados muestran dichos niveles de reducción sobre diferentes baterías de problemas.

El método exacto no presenta un mal comportamiento con instancias con pocos recursos y de hasta 100 pedidos. Sin embargo, el carácter exponencial del número de nodos y arcos respecto al número de recursos del problema, imposibilita la aplicación de este método cuando se aumentan los mismos. Este hecho fuerza la necesidad de implementar procedimientos aproximados. En primer lugar se presenta un GRASP cuyos resultados son válidos en tiempo aunque con un error mejorable para problemas de tamaño grande. Un algoritmo de Búsqueda Tabú corrige las deficiencias presentadas por el procedimiento GRASP. Este algoritmo se valida con diferentes baterías de problemas que aumentan la “estrechez” de la actividad de los pedidos en el horizonte temporal. Incluso sobre estas baterías, el algoritmo presenta un comportamiento válido. Además de ello, la característica más sobresaliente de la búsqueda tabú es que con el aumento en los recursos del problema disminuye el error de los resultados que proporciona. Por todo ello, puede concluirse que este enfoque basado en búsqueda tabú es un método apropiado para la resolución del problema en este escenario.

- **Escenario IB**

Se estudia el problema PDP con un número ilimitado de vehículos. Se presenta, por tanto, un enfoque en el que únicamente la planta de producción impone

limitación en el procesamiento de los pedidos. El problema PDP en el Escenario IB recibe a su vez dos enfoques distintos:

- Caso general: El valor de los pedidos puede ser cualquiera
- Caso particular: Mismo valor para todos los pedidos

Caso general: Este caso se asimila completamente al problema FSP. El problema se resuelve con un algoritmo de flujo a coste mínimo sobre una nueva construcción del grafo asociado al problema. Dicha construcción reduce el tamaño del grafo respecto a otras implementaciones realizadas en la literatura.

Caso particular: En este caso el objetivo del problema puede verse como el de maximizar el número de pedidos servidos. Este objetivo se muestra interesante porque puede llevar probablemente a la aparición de soluciones óptimas complementarias que se diferencian en el número de vehículos utilizados. El problema se centra entonces en el cálculo de la solución que minimiza el número de vehículos usados. La resolución de este problema recibe un enfoque exacto y un enfoque aproximado.

El método exacto de resolución está basado en un procedimiento iterativo de construcción de un grafo a partir del método exacto descrito en el Escenario IA. El método se muestra ineficiente con problemas de tamaño grande y con un número alto en los recursos de producción.

El enfoque heurístico implementado se muestra como uno de los más satisfactorios de la investigación. Está basado en un procedimiento de ramificación y acotación con un recorrido en anchura del árbol de exploración, en el que el número de niveles a explorar se considera un parámetro. Los experimentos realizados corroboran la validez del método. En tiempos de ejecución prácticamente despreciables se consiguen las soluciones óptimas de los problemas.

10.1.2. Escenario II

El Escenario II presenta el problema PDP con una planta de producción y ventanas temporales de entrega. El estudio en este escenario comienza con la formulación del problema mediante un modelo de programación entera. La resolución de forma exacta del problema y, en general, la obtención de buenas soluciones, conlleva mayor dificultad que en el Escenario I. El método exacto implementado se basa en la idea expresada para el método exacto del Escenario IA, aunque en este caso, la

filosofía del proceso de construcción del grafo tiene que recoger los diferentes instantes en el que puede comenzar el procesamiento de un pedido.

En este escenario se han introducido como métodos de resolución de problemas los algoritmos genéticos. Dos enfoques diferentes se presentan para el problema. El primero se trata de un algoritmo convencional que no explota las características del problema. El segundo enfoque si provee un algoritmo que aprovecha el estudio del problema PDP realizado hasta ese instante. Ello se traduce en la aplicación de técnicas de cálculo de flujo a coste mínimo sobre grafos que garantizan el aprovechamiento óptimo de los recursos en situaciones dadas. Los experimentos realizados confirman la conveniencia del uso de este segundo enfoque, si bien la mayor complejidad en el cálculo del *fitness* de cada individuo lleva a tiempos de ejecución mayores.

Con objeto de realizar un segundo análisis sobre la validez del algoritmo genético, se presenta un nuevo método basado en búsqueda tabú. Esta implementación aprovecha las características de la búsqueda tabú diseñada en el Escenario IA del problema. Los experimentos computacionales ponen de manifiesto un comportamiento similar de los dos enfoques con respecto al error medio, aunque para ello el algoritmo de búsqueda tabú necesite de más iteraciones. Sin embargo, respecto a los tiempos de computación, el algoritmo de búsqueda tabú se muestra más adecuado para la resolución del problema.

10.1.3. Escenario III

En el Escenario III se ha estudiado el problema PDP con varias plantas de producción e instantes fijos de entrega. En el estudio de los escenarios con varias plantas se ha seguido una metodología que intenta aprovechar el estudio realizado en los escenarios con una sola planta. Ello se pone de manifiesto en la formulación y definición de estrategias de solución.

Tras definir un modelo de programación entera para el problema, en el estudio de este escenario se considera en primer lugar un caso especial que puede ser resuelto en tiempo polinomial. Este caso responde a situaciones reales en este tipo de sistemas. La solución proporciona una cota superior del óptimo del problema para el caso general, y por ello, es utilizado como base de inicio en el procedimiento heurístico propuesto para la resolución del problema.

La calidad de las soluciones proporcionadas por el procedimiento heurístico se ha medido respecto a las soluciones óptimas proporcionadas por un método exacto, basado en un grafo de estados admisibles. Respecto a escenarios anteriores, el

proceso de construcción de este grafo aumenta considerablemente la dificultad para la creación de estados admisibles. Los resultados computacionales llevados a cabo indican que la heurística encuentra soluciones de buena calidad y puede ser usada con cierta garantía con problemas de tamaño mayor.

10.1.3. Escenario IV

Finalmente, como último escenario de estudio del problema PDP se presenta el escenario genérico del problema, en el que pueden incluirse el resto de los escenarios anteriores como caso particulares del mismo. El estudio de este escenario se centra aún más en la aplicación de técnicas ya utilizadas en escenarios anteriores y que obtuvieron resultados satisfactorios. En concreto, para el diseño de un algoritmo genético se alude a uno de los algoritmos genéticos descritos en el Escenario II del problema y al procedimiento heurístico definido en el Escenario III. Por otro lado, también se implementa una variante del método exacto del Escenario III, que admite el uso de ventanas temporales, y que por tanto, aumenta aún más la complejidad de obtención de soluciones.

Los resultados vuelven a mostrar un buen comportamiento de las técnicas heurísticas utilizadas, aunque los experimentos en este escenario estuvieron más limitados que en los anteriores.

10.2. FUTURAS LÍNEAS DE TRABAJO

Como líneas de investigación a medio plazo, esto es, continuación de la línea llevada a cabo en esta tesis, se apuntan los siguientes campos de trabajo:

- Aplicación de otras metaheurísticas para la resolución del problema PDP.
- Estudio del problema PDP con demanda de pedidos dinámica.

Otras líneas de investigación relacionadas con los problemas vistos en esta tesis, y como continuación de la línea iniciada en la secuenciación de trabajos con intervalos fijos de proceso, serían el estudio de los siguientes problemas, los cuales están íntimamente relacionados:

- Estudio del problema FSP con varias clases de máquinas [Kroon et al, 1995].

- Estudio del problema de asignación de *k*-vías (versión no determinista del problema FSP con varias clases de máquinas) [Faigle et al, 1999] [Brucker y Norman, 1994].

Anexo I

Generación de problemas

Índice

I.1. INTRODUCCIÓN

En este anexo se describe el diseño de las diferentes baterías de problemas utilizadas en la tesis. El parámetro fundamental en el diseño de los problemas ha sido el grado de simultaneidad de los pedidos durante el horizonte temporal de la planificación. Previamente a la descripción de este cálculo sobre cada uno de los escenarios del problema, describimos la asignación de los datos de un pedido.

I.2. ASIGNACIÓN DE LOS DATOS DE UN PEDIDO

En general, los valores asignados a cada uno de los datos de un pedido han sido obtenidos dentro de los siguientes intervalos:

Parámetro	Intervalo
Tiempo de producción	[1,5]
Tiempo de envío	[1,10]
Tiempo de descarga	[1,2]
Tiempo de vuelta	[1,10]
Amplitud de ventana temporal	[1,5]
Valor del pedido	[30,100]
Penalización por unidad de retraso o adelanto	[0,2]

Tabla I.1: Intervalos de los datos asignados a un pedido

La discretización del tiempo realizada sobre el horizonte de planificación asigna a cada periodo un valor equivalente a algunos minutos de tiempo. Sería del orden de 3 minutos por periodo de tiempo para el caso de planificaciones temporales de un día, producción simplificada y entorno geográfico de la actividad reducido. Ello lleva a una media de 8 a 10 minutos de tiempo de fabricación, y 15 a 20 minutos de tiempo de envío por pedido. De cualquier modo, la asignación del tiempo real de cada periodo puede ser adaptada a otros valores.

Esta asignación confiere a la fase de distribución (envío, descarga y vuelta) un 80% aproximadamente del tiempo de actividad total del pedido, lo cual confiere a esta fase una importancia mayor respecto a la disponibilidad de recursos.

Respecto a la amplitud de las ventanas temporales, la complejidad para la obtención de soluciones óptimas en los escenarios donde aparecen imposibilita el estudio del problema con valores mayores, aunque las propias características del

problema determinan ventanas temporales de amplitud reducida, debido fundamentalmente a las características de la gestión del producto.

Otro tipo de dato asignado a cada pedido es el valor del mismo, el cual se determina como un beneficio económico obtenido por atender el pedido. Para este dato se ha usado un intervalo amplio de valores ([30,100]), en los que una unidad poseería un valor estimado en función de la actividad concreta en la que se desarrolle la planificación. En los escenarios con una planta, dentro del valor del pedido se incluye el coste de la distribución del mismo.

I.3. CÁLCULO DE LOS GRADOS DE SIMULTANEIDAD

En este apartado se presenta el cálculo de los promedios de simultaneidad que se realiza en cada escenario. La notación que se utiliza en todos los escenarios es la siguiente:

L_t^P = Grado de simultaneidad en la fase de producción en el periodo *t*

L_t^D = Grado de simultaneidad en la fase de distribución en el periodo *t*

L_m^P = Grado medio de simultaneidad en la fase de producción

L_m^D = Grado medio de simultaneidad en la fase de distribución

L^P = Mayor grado de simultaneidad en la fase de producción

L^D = Mayor grado de simultaneidad en la fase de distribución

❑ ESCENARIO I

El cálculo de los grados de simultaneidad es el mismo tanto en el Escenario IA como en el Escenario IB. A continuación describimos el procedimiento:

1. Tras una ordenación previa de los pedidos por instante de inicio s_i, se calcula el instante de comienzo de la planificación (instante de comienzo del primer pedido) y el instante de finalización (instante de finalización del último trabajo). Denotemos por *S* y *F* esos dos instantes. Por tanto, el intervalo [*S* , *F*] contiene la actividad de todos los pedidos solicitados.

2. El procedimiento de cálculo de los promedios L y L^D , $t = S...F$, chequea si las fases de producción y distribución de cada pedido se producen durante el periodo *t.* Para ello se tienen en cuenta los instantes de inicio y finalización de cada fase:

   s_i : instante de inicio de la fase de producción del pedido *i*

   l_i : instante de inicio de la fase de distribución del pedido *i*; por tanto, l_i-1 es el instante de finalización de la fase de producción. Para el caso definido en el Escenario IB, el instante de inicio de la fase de distribución se corresponde también con s_i

   f_i: instante de finalización de la fase de distribución del pedido *i*

El procedimiento de cálculo es el siguiente:

Para $t = S$ hasta F
  Para $i = 1$ hasta n
    Si $s_i <= t$ & $l_i > t$ **entonces**
      $L^P{}_t = L^P{}_t + 1$
    Fin Si
    Si $l_i <= t$ & $f_i >$ t **entonces**
      $L^D{}_t = L^D{}_t + 1$
    Fin Si
  Fin Para
Fin Para

Los parámetros que definen el grado de solapamiento de los pedidos se obtienen como la media y el máximo de los parámetros calculados anteriormente:

$$L_m^P = \frac{\sum_{t=S}^{F} L_t^P}{F-S}; \quad L_m^D = \frac{\sum_{t=S}^{F} L_t^D}{F-S}; \; L^P = \text{máximo}\{L^P{}_t\}; \; L^D = \text{máximo}\{L^D{}_t\};$$

- **ESCENARIO II**

El cálculo de los grados de simultaneidad para el caso en el que se agregan ventanas temporales, es idéntico al descrito para el escenario anterior. Es decir, los cálculos de simultaneidad se realizan respecto a los instantes ideales de comienzo y finalización de las fases del pedido.

❑ ESCENARIO III

El Escenario III presenta el problema PDP con varias plantas de producción. Aquí se hace necesario un nuevo diseño del cálculo de los grados de solapamiento, puesto que la actividad de los pedidos es variable y puede producirse en diferentes plantas. En concreto, la parte variable es la fase de distribución, la cual depende de la planta desde donde se produce el pedido y de la planta a la que regresa el vehículo.

Para establecer un paralelismo con el cálculo realizado en los dos escenarios anteriores, el mecanismo que se sigue en este escenario es el de suponer la planificación de los pedidos teniendo en cuenta cada una de las plantas de forma independiente. Esto significa que, para cada planta, se plantea la situación del Escenario I (todos los pedidos se producen en la misma planta a la que también regresan siempre los vehículos tras servir el pedido). De esta forma, podemos aplicar los cálculos del Escenario I sobre cada una de las plantas. Para ello definimos la siguiente notación:

$L_t^{P(k)}$ = Grado de simultaneidad en la fase de producción en el periodo *t* sobre la planta *k*

$L_t^{D(k)}$ = Grado de simultaneidad en la fase de distribución en el periodo *t* sobre la planta *k*

$L_m^{P(k)}$ = Grado medio de simultaneidad en la fase de producción sobre la planta *k*

$L_m^{D(k)}$ = Grado medio de simultaneidad en la fase de distribución sobre la planta *k*

El cálculo de estos cuatros parámetros es análogo al realizado en los escenarios previos. Tan sólo se necesita tener en cuenta los instantes de comienzo y finalización de las fases de producción y distribución sobre cada planta:

s_{ik} : instante de inicio de la fase de producción del pedido *i* en la planta *k*

l_{ik} : instante de inicio de la fase de distribución de *i* desde la planta *k*

f_{ik} : instante de finalización de la fase de distribución de *i* en la planta *k*

Una vez calculados los índices $L_m^{P(k)}$ y $L_m^{D(k)}$ para k=1...K (número de plantas), los grados medios de simultaneidad se obtienen como:

$$L_m^P = \frac{\sum_{k=1}^{K} L_m^{P(k)}}{K}; \qquad L_m^D = \frac{\sum_{k=1}^{K} L_m^{D(k)}}{K};$$

En este escenario se omite el cálculo de L^P y L^D porque se entiende que con el cálculo realizado, el valor de esos parámetros no reflejaría el significado de los mismos.

- **ESCENARIO IV**

Como ocurría en el Escenario II respecto al Escenario I, en el Escenario IV se realiza el mismo procedimiento que en el Escenario III, atendiendo exclusivamente al instante ideal de comienzo sobre cada planta.

Anexo II

Contenido del CD-ROM

Índice

II. . INTRODUCCIÓN

En este anexo se describe el contenido del CD-ROM que se adjunta con el documento de la tesis. El CD contiene todos los archivos de entrada, algunos archivos de salida y una base de datos con los resultados de todos los experimentos realizados en cada escenario. También se adjunta el código de programación de todos los métodos desarrollados. Por último, se incluye, en formato PDF, la propia memoria de la tesis doctoral. En el anexo se describe, además de la estructura general de CD, el formato de todos los archivos.

II. . ESTRUCTURA ENERAL DEL CD

Denominando "CD-ROM" a la unidad de CD, la estructura de directorios se muestra en la figura II.1.

- **C E**

En la carpeta "Entrada" se encuentran todas las baterías de problemas utilizadas en cada escenario. Cada problema está contenido en un archivo de entrada contiene cuyo nombre responde al siguiente formato:

{ *ip de e re a*}{ *ama* }_{ *mer* }[_{ *a as*}{ *e s*}].in

ip de e re a : Hace referencia al tipo de fecha de entrega para el problema:

F: Fija (Escenario I y III)

V: Ventana temporal (Escenario II y IV)

ama : Hace referencia al número de pedidos del problema. Cada tamaño se identifica por un número entero secuencial, asignando el valor 1 al tamaño menor. El número de pedidos para cada valor es dependiente del escenario.

mer : Número de la instancia. Existen 10 instancias para cada tamaño en todas las baterías.

a as : Número de plantas del problema. Se especifica en los problemas de los escenarios III y IV.

e s : Número de vehículos en el problema. Se especifica sólo en los escenarios con múltiples plantas, pues en los datos del problema hay que especificar el número de vehículos que existen inicialmente en cada planta.

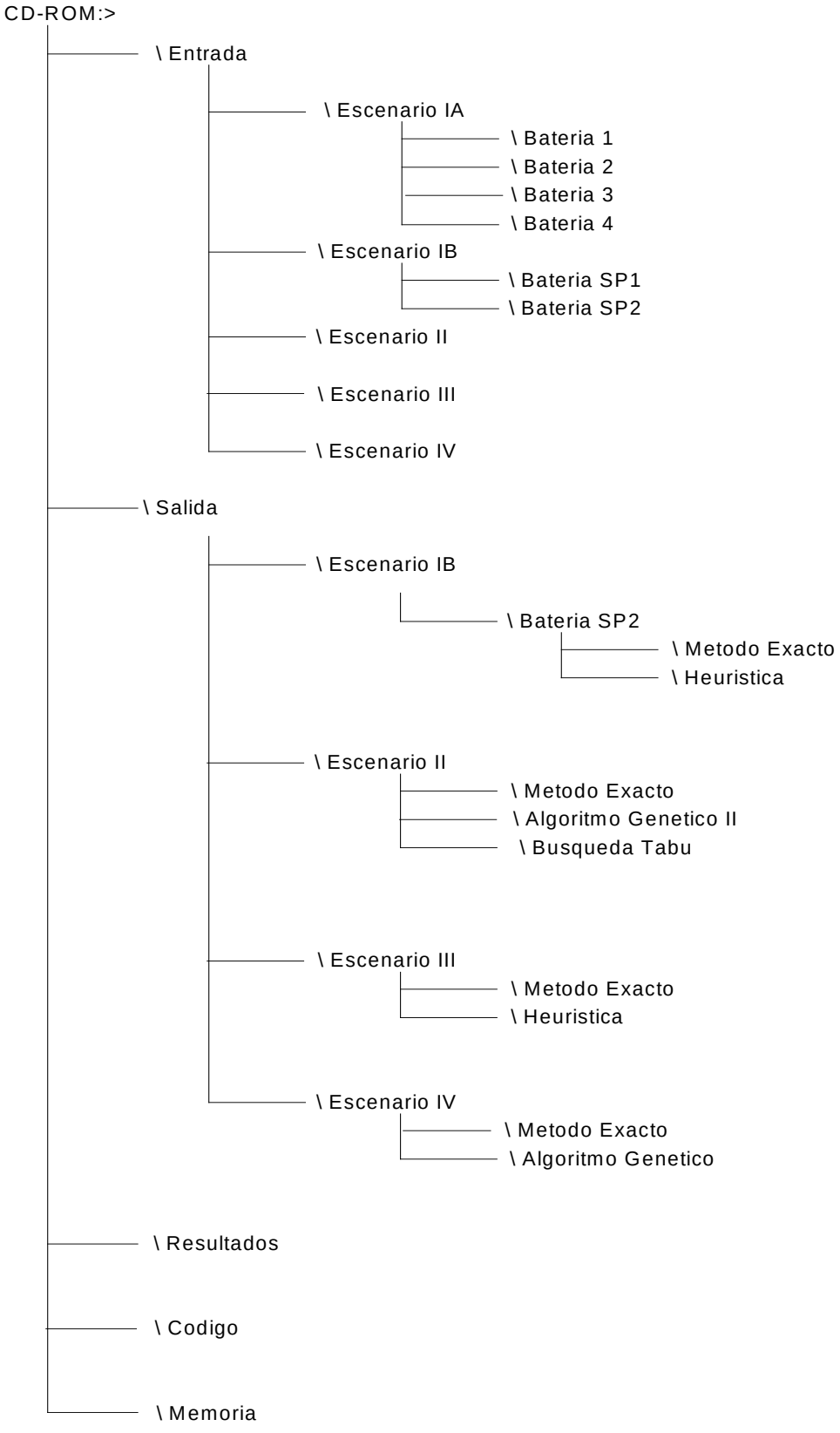


Figura II.1: Estructura del CD

Ejemplos:

"F2_04.in": Instancia número 4 de los problemas de tamaño 2 con instantes fijos de entrega (Escenario I).

"V1_03.in": Instancia número 3 de los problemas de tamaño 1 con ventanas temporales de entrega (Escenario II).

"F4_06_23.in": Instancia número 6 de los problemas de tamaño 4 con instantes fijos de entrega, 2 plantas y 3 vehículos (Escenario III).

"V1_10_32.in": Instancia número 10 de los problemas de tamaño 1 con ventanas temporales de entrega, 3 plantas y 2 vehículos (Escenario IV).

- **C S**

En la carpeta "Salida" se encuentran los archivos resultado de la aplicación de algunos de los métodos de resolución implementados. Cada archivo de salida contiene el resultado de cada archivo de entrada. El formato del nombre es el siguiente:

{ *ip e re a*}{ *ama* }_{ *mer* }_[*a as*]{ *e s*}[*apa idad*][*ar i ipa*].out

ip de e re a ama mer Significado equivalente al descrito en el nombre de los archivos de entrada.

e s: Número de vehículos con el que se resolvió el problema.

a as : Número de plantas del problema. Este dato no se especifica en los métodos de resolución de los escenarios I y II.

apa idad : Capacidad de producción con la que se resolvió el problema. Este dato no se especifica en los métodos de resolución de los escenarios III y IV, puesto que la capacidad es igual a 1 para todas las plantas en todas las ejecuciones.

ar i ipa: Clave que especifica si el vehículo participó en el proceso productivo:

S: El vehículo participa.

N: El vehículo no participa.

Este clave es también opcional puesto que en los escenarios III y IV no aparece, ya que en todos los experimentos realizados sobre estos escenarios se supuso que el vehículo no participaba en el proceso productivo.

Ejemplos:

“F2_04_42N.out”: Instancia número 4 de los problemas de tamaño 2 con instantes fijos de entrega (Escenario I) resuelta con 4 vehículos, capacidad igual a 2 y sin la participación del vehículo en el proceso productivo. El archivo de entrada que contiene los datos del problema se denominaría “F2_04.in”.

“V1_10_32.out”: Instancia número 10 de los problemas de tamaño 1 con ventanas temporales de entrega, 3 plantas y 2 vehículos (Escenario IV). El archivo de entrada sería “V1_10_32.in”.

- **C C**

La carpeta Código contiene el código de todos los métodos de resolución empleados en la tesis. Cada método se encuentra en un archivo cuyo nombre especifica claramente el método de resolución y el escenario del problema. El formato es:

Esc{ *mer de es e ari* }_{ *m re de m d* }.bas

- **C R**

En la carpeta resultados se encuentra la base de datos *A ess* “Datos.mdb” que almacena todos los resultados de los problemas. Los resultados de cada método de resolución se encuentran en cada una de las tablas de la base de datos. El nombre de cada tabla es el mismo que identifica cada archivo de código definido anteriormente.

- **C M**

Esta carpeta contiene la memoria de la tesis en formato *pd* . El nombre del archivo es “ memoria.pdf ”.

II. . FORMATO DE LOS ARC I OS DE ENTRADA

En este apartado se describe el formato de los archivos de entrada de cada escenario.

❑ E IA IB

El formato de los archivos de entrada de los escenarios IA y IB es el mismo. La estructura de cada archivo es la siguiente:

FICHERO ENTRADA: {*nombre del fichero de entrada*}

ESCENARIO: {*Número de escenario*}
NUMERO PEDIDOS: *n*

PEDIDOS	s	l	f	w	tc	td	ti	tv
PEDIDO 1:	s_1	l_1	f_1	w_1	tp_1	tu_1	ti_1	tv_1
PEDIDO 2:	s_2	l_2	f_2	w_2	tp_2	tu_2	ti_2	tv_2
.....								
PEDIDO *n*:	s_n	l_n	f_n	w_n	tp_n	tu_n	ti_n	tv_n

Figura II.2: Formato general de los archivos de entrada en los escenarios IA y IB

- **E II**

El formato de los archivos de entrada es el siguiente:

FICHERO ENTRADA: {*nombre del fichero de entrada*}

ESCENARIO: {*Número de escenario*}
NUMERO PEDIDOS: *n*

PEDIDOS	s	l	f	I-	I+	w	w-	w+	tc	td	ti	tv
PEDIDO 1:	s_1	l_1	f_1	s_1-a_1	b_1-s_1	W_1	w^-_1	w^+_1	tp_1	tu_1	ti_1	tv_1
PEDIDO 2:	s_2	l_2	f_2	s_2-a_2	b_2-s_2	W_2	w^-_2	w^+_2	tp_2	tu_2	ti_2	tv_2
.....												
PEDIDO *n*:	s_n	l_n	f_n	s_n-a_n	b_n-s_n	W_n	w^-_n	w^+_n	tp_n	tu_n	ti_n	tv_n

Figura II.3: Formato general de los archivos de entrada en el Escenario II

- **E III**

El formato de los archivos de entrada es el siguiente:

FICHERO ENTRADA: {*nombre del fichero de entrada*}

ESCENARIO: {*Número de escenario*}
NUMERO PLANTAS: m
NUMERO VEHICULOS: V
NUMERO PEDIDOS: n

PEDIDOS	e	w	tc	td	ti	tv			
PEDIDO 1:	e_1	w_1	tp_1	tu_1	ti_{11}	tv_{11}	...	ti_{1m}	tv_{1m}
PEDIDO 2:	e_2	w_2	tp_2	tu_2	ti_{21}	tv_{21}	...	ti_{2m}	tv_{2m}
.....									
PEDIDO n:	e_n	w_n	tp_n	tu_n	ti_{n1}	tv_{n1}	...	ti_{nm}	tv_{nm}

$v_1 \ldots v_m$

Figura II.4: Formato general de los archivos de entrada en el Escenario III

❑ E I

El formato de los archivos de entrada es el siguiente:

FICHERO ENTRADA: {*nombre del fichero de entrada*}

ESCENARIO: {*Número de escenario*}
NUMERO PLANTAS: *m*
NUMERO VEHICULOS: *V*
NUMERO PEDIDOS: *n*

PEDIDOS	e	I-	I+	w	w+	w-	tc	td	ti	tv		
PEDIDO 1:	e_1	e_1-A_1	B_1-e_1	W_1	w^-_1	w^+_1	tp_1	tu_1	ti_{11}	tv_{11}	...ti_{1m}	tv_{1m}
PEDIDO 2:	e_2	e_2-A_2	B_2-e_2	W_2	w^-_2	w^+_2	tp_2	tu_2	ti_{21}	tv_{21}	...ti_{2m}	tv_{2m}
.....												
PEDIDO *n*:	e_n	e_n-A_n	B_n-e_n	W_n	w^-_n	w^+_n	tp_n	tu_n	ti_{n1}	tv_{n1}	...ti_{nm}	tv_{nm}

Figura II.5: Formato general de los archivos de entrada en el Escenario IV

La notación que aparece en los diferentes archivos de entrada es la notación referida en los diferentes escenarios de la tesis.

II. . FORMATO DE LOS ARC I OS DE SALIDA

En este apartado se describe el formato de los archivos de salida que aparecen en el CD-ROM.

❑ E IB

La estructura de los archivos es la siguiente:

```
FICHERO: {nombre del fichero de entrada}

Escenario: {Número de escenario}
Método:   {Nombre de método}

Número de pedidos:  n
Número de vehículos: 0
Capacidad: C
Vehículo-Producción: {Si|No}

Función objetivo:    {valor de la función objetivo}
Pedidos servidos:    {número de pedidos servidos}
Vehículos usados:    {número de vehículos utilizados}
Tiempo de ejecución: {tiempo de ejecución}

PEDIDOS     s    l    f
  i         s_i  l_i   f_i
  j         s_j  l_j   f_j
  .......

VEHÍCULOS
Vehículo 1: {lista de pedidos 1}
Vehículo 2: {lista de pedidos 2}
.....
Vehículo v: {lista de pedidos v}
```

Figura II.7: Formato general de los archivos de salida en el Escenario IB

Respecto al formato de los archivos del Escenario IA, en el Escenario IB se agrega el número de vehículos utilizados. En el campo “Número de Vehículos” aparece un 0 en todo los problemas, aunque el número de vehículos es ilimitado en este escenario. Al final del archivo se apunta la lista de pedidos que ha distribuido cada vehículo. Cada pedido se referencia por un número.

❑ E II

La estructura de los archivos en el Escenario II es similar a la estructura de los archivos en el Escenario IA (figura II.6).

```
FICHERO: {nombre del fichero de entrada}

Escenario: {Número de escenario}
Método:   {Nombre de método}

Número de pedidos:   n
Número de vehículos: V
Capacidad: C
Vehículo-Producción: {Si|No}

Función objetivo:    {valor de la función objetivo}
Pedidos servidos:    {número de pedidos servidos}
Tiempo de ejecución: {tiempo de ejecución}

PEDIDOS     s        l        f        w
  i         s_i+x_i  l_i+x_i  f_i+x_i  w_i
  j         s_j+x_j  l_j+x_j  f_j+x_j  w_j
  .......
```

Figura II.8: Formato general de los archivos de salida en el Escenario II

Sin embargo, en los archivos de salida del Escenario II la lista de pedidos seleccionados incluye los instantes que definen el desarrollo de la actividad del pedido, datos que son variables cuando existen ventanas temporales. Por ello, llamando $_i$ al número de instantes de retraso (valor positivo) o adelanto (valor negativo) del pedido *i* respecto a su instante de entrega ideal, $s_i + _i$ expresa el instante de inicio de la fase producción. De forma análoga, $_i + _i$ y $_i + _i$ representan los instantes de comienzo y finalización, respectivamente, de la fase de distribución.

❑ E III

La estructura de los archivos en el Escenario III es la siguiente:

FICHERO: {*nombre del fichero de entrada*}

Escenario: {*Número de escenario*}
Método: {*Nombre de método*}

Número de pedidos: *n*
Número de plantas: *m*
Número de vehículos: *V*
Vehículo-Producción: {Si|No}

Función objetivo: {*valor de la función objetivo*}
Pedidos servidos: {*número de pedidos servidos*}
Tiempo de ejecución: {*tiempo de ejecución*}

PEDIDOS	O	D	s	l	f	w	*	Coste
i	*O(i)*	*D(i)*	$s_{iO(i)}$	$l_{iO(i)}$	$f_{iD(i)}$	w_i	y_i	c_i
j	*O(j)*	*D(j)*	$s_{jO(j)}$	$l_{jO(j)}$	$f_{jD(j)}$	w_j	y_j	c_j
.....								

Planta 1: *i* ... *r*
Inicio 1: s_{i1} ... s_{r1}
Fin 1: l_{i1}-1 ... l_{r1}-1
.....

Planta *m*: *j* ... *t*
Inicio *m*: s_{j1} ... s_{t1}
Fin *m*: l_{j1}-1 ... l_{t1}-1

Vehículo 1: *i* ... *t*
Pl Inic 1: *O(i)* ... *O(t)*
Pl Fin 1: *D(i)* ... *D(t)*
Inicio 1: $l_{iO(i)}$ ... $l_{tO(t)}$
Fin 1: $f_{iD(i)}$ ... $f_{tD(t)}$
.....

Vehículo *V*: *j* ... *r*
Pl Inic V: *O(j)* ... *O(r)*
Pl Fin *V*: *D(j)* ... *D(r)*
Inicio *V*: $l_{jO(j)}$ ... $l_{rO(r)}$
Fin *V*: $f_{jD(j)}$ ... $f_{rD(r)}$

Figura II.9: Formato general de los archivos de salida en el Escenario III

En la descripción del formato de los archivos de salida de este escenario, se ha utilizado una notación no descrita en el documento y que comentamos a continuación:

(i) : Planta en la que procesa el pedido i

(i) : Planta a la que regresa el vehículo tras servir el pedido i

$_i$: Número que identifica al vehículo que sirve el pedido i

$_i$: Coste de servir el pedido i; $_i = d_{iO(i)} + d_{iD(i)}$

Como se observa en el formato del archivo, se ha pretendido registrar la mayor información posible. Por ello se describe la actividad en cada una de las plantas y de cada uno de los vehículos.

❑ E I

El formato en este escenario es similar al descrito en el escenario anterior. Lo único que añade es el número de periodos de retraso (valor positivo) o adelanto (valor negativo) sobre el instante ideal de cada pedido servido. Este dato se presenta en la columna cuyo título de cabecera es "i". El valor representado en ese parámetro se ha definido de forma semejante al definido en el Escenario II, es decir, mediante la notación $_i$. Este valor se suma a los instantes ideales de comienzo y finalización de los pedidos.

FICHERO: {*nombre del fichero de entrada*}

Escenario: {*Número de escenario*}
Método: {*Nombre de método*}

Número de pedidos: *n*
Número de plantas: *m*
Número de vehículos: *V*
Vehículo-Producción: {Si|No}

Función objetivo: {*valor de la función objetivo*}
Pedidos servidos: {*número de pedidos servidos*}
Tiempo de ejecución: {*tiempo de ejecución*}

PEDIDOS	O	D	i	s	l	f	w	*	Coste
i	$O(i)$	$D(i)$	x_i	$s_{iO(i)}+x_i$	$l_{iO(i)}+x_i$	$f_{iD(i)}+x_i$	w_i	y_i	c_i
j	$O(j)$	$D(j)$	x_j	$s_{jO(j)}+x_j$	$l_{jO(j)}+x_j$	$f_{jD(j)}+x_j$	w_j	y_j	c_j
.....									

Planta 1:	i	...	r
Inicio 1:	$s_{i1}+x_i$	...	$s_{r1}+x_r$
Fin 1:	$l_{i1}-1+x_i$	...	$l_{r1}-1+x_r$
.....			
Planta m:	j	...	t
Inicio m:	$s_{j1}+x_j$	...	$s_{t1}+x_t$
Fin m:	$l_{j1}-1+x_j$	...	$l_{t1}-1+x_t$

Vehículo 1:	i	...	t
Pl Inic 1:	$O(i)$	...	$O(t)$
Pl Fin 1:	$D(i)$	...	$D(t)$
Inicio 1:	$l_{iO(i)}+x_i$	...	$l_{tO(t)}+x_t$
Fin 1:	$f_{iD(i)}+x_i$	...	$f_{tD(t)}+x_t$
.....			
Vehículo V:	j	...	r
Pl Inic V:	$O(j)$	...	$O(r)$
Pl Fin V:	$D(j)$	...	$D(r)$
Inicio V:	$l_{jO(j)}+x_j$	...	$l_{rO(r)}+x_r$
Fin V:	$f_{jD(j)}+x_j$	...	$f_{rD(r)}+x_r$

Figura II.10: Formato general de los archivos de salida en el Escenario IV

REFERENCIAS

Ahuja R.K., Magnanti T.L., Magnanti J. and Orlin J.B. (1993), Network Flows: theory, algorithms, and applications. Prentice-Hall International, Englewood Cliffs, N.J.

Arthanari T.S. and Ramamurthy, K.G. (1971), An extension of two machines sequencing problem. *Opsearch*, 8, 10-22.

Adler L.B. (1993), BPSS: A scheduling support system for the packaging industry. J. *Operations Research* 41, 641-648.

Aiex R. M., Resende M.G.C., Pardalos P.M. and Toraldo G. (2000), GRASP With Path Relinking For The Three-Index Assignment, *IPDPS Workshops*.

Arkin E.M. and Silverberg E.B. (1987), Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18, 1-8.

Baker K.R. and Scudder G.D. (1990), Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38, 22-36.

Bertolissi, E. (2000), Heuristic algorithm for scheduling in the no-wait flow-shop. *Journal of Materials Processing Technology*, 107, 459-465.

Bertsekas D. and Tseng P. (1988), Relaxation methods for minimum cost for ordinary and generalized network flow problems. *Operations Research*, 36, 93-114.

Brah S.A. and Hunsucker J.L. (1991), Brach and bound algorithm for the flow shop with multiple processors. *European Journal of Operational Research* 51, 88-99.

Bratley P., Florian M. and Robillard P. (1975), Scheduling with earliest start and due date constraints on multiple machines. *Naval Research Logistics* 22 (1), 165-173.

Brucker P. and Nordmann L. (1994), The k-track assignment problem. *J. Computing*, 52, 97-122

Chen B. (1995), Analysis of Classes of Heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the operational research society* 46, 234-244.

Chen Z. and Powell W.B. (1999), A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. *European Journal of Operational Research* 116, 220-232.

Cheng T.C.E. and Sin C.C.S. (1990), A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47, 271-292

Chuzhoy J. and Ostrovsky R. (2001), Aproximation algorithms for the job interval selection problem and related scheduling problems. *FOCS 2001- IEEE proceedings of the 42nd Annual Symposium on Foundations of Computer Science* held on October, 14-17.

Diaz A., Glover F. Ghaziri H.M., Gonzalez J.L., Laguna M., Moscato P. and Tseng F.T. (1996), Optimización heurística y Redes neuronales en Dirección de operaciones e Ingeniería. Editorial Paraninfo. ISBN: 84-283-2269-4, 26-27.

Drees L.D. and Wilhelm W.E. (2001), Scheduling experiments on a nuclear reactor using mixed integer programming. *Computers and Operations Research* 28, 1013-1037.

Eguia I. (1996), Programación y control de la producción en procesos semicontinuos. Métodos y algoritmos de resolución. Tesis doctoral presentada en la Escuela Superior de Ingenieros de la Universidad de Sevilla.

Emmons H. (1987), Scheduling to a common Due Date on parallel common processors. *Naval Research Logistic* 34, 803-810.

Faigle U., Kern W., Nawijn M. (1999), A greedy On-line algorithm for the k-track assignment problem. *Journal of Algorithms*, 31, 196-210

Feo T., Venkatraman K. and Bard J. (1991), A GRASP for a difficult single machine scheduling problem, *Computers and Operations Research*, 18.

Feo T.A. and M.G.C. Resende (1995), Greedy Randomized Adaptative Search Procedures, *Journal of Global Optimization* 6, pp. 109-133.

Fischetti M., Martello S. and Toth P. (1992), Approximation algorithms for fixed job schedule problems, *Operations Research* 40/1, pp. S96-S108.

Frangioni A. (2001), *http://www.di.unipi.it/di/groups/optimize/Software/index.html*

Gabrel V. (1995), Scheduling jobs within time windows on identical parallel machines: New model and algorithms. *European Journal of Operations Research*, 83, 320-329.

Garey M.R. and Johnson D.S. (1979), Computers and Intractability: A guide to the theory of NP-Completeness. W.H. Freeman, San Francisco.

Gertsbakh I. and Stern H. (1978), Minimal Resources for Fixed and Variable Job Schedules. *Operations Research*, 26 (1), 68-85.

Glover F. (1989), Tabu search. Part I. *ORSA Journal on Computing*, 1, 190-206.

Glover F. (1990), Tabu search. Part II. *ORSA Journal on Computing*, 2, 4-32.

Glover F. and Laguna M, (1997). Tabu Search, Kluwer Academic Publishers, Dordrecht.

Glover F., Taillard E. and Werra D. (1993), A user's guide to tabu search. *Annals of Operations Research*, 41, 3-28.

Goldberg, D. E. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning. New York, NY: Addison-Wesley.

Graham R.L., Lawler E.L., Lenstra J.K. and Rinnoy Kan A.H.G. (1979), Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, 5, 287-326.

Gupta J.N.D. (1988), Two stage hybrid flow-shop scheduling problem. *Journal of the operational research* 29/7, 1489-1502.

Gupta J.N.D., Hariri A.M.A. and Potts C.N. (1997), Scheduling a Two-Stage Flow Shop with Parallel Machines at the First Stage, *Oper Res*, 69: pp. 171-191.

Gupta, J. N. D., and Tunc, E. A. (1991), Schedules for a Two-Stage Hybrid Flowshop with Parallel Machines at the Second Stage, *International Journal of Production Research*, Vol. 29, No. 7, pp. 1489-1502.

Hall N. (1986), Single and Multi-Processor models for minimizing completion time variance. *Naval Research Logistic*, 33, 49-54.

Hall N.G. and Sriskandarajah C. (1996), A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44, 510-525.

Haouari M. and M'Hallah R. (1997), Heuristic algorithms for the two-stage hynrid flowshop problem. *Operations research letters* 21, 43-53.

Hashimoto A. and Stevens J. (1971), Wire routing by optimizing channel assignments within apertures, in: *Proceedings of the 8th Design Automation Workshop*, pp. 155-169.

Hiraishi K, Levner E. and Vlach M. (2002), Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs. *Computers and Operations Research*, 29, 841-848.

Holland J. (1975), Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor.

Huang W. and Li S. (1998), A two-stage hybrid flowshop with uniform machines and setup times. *Mathematical Comput. Modeling* 27, nº2, 27-45.

Kaufmann, M. (1991), Fundations of Genetic Algorithms. Edited by Gregory J.E. Rawlins, San Mateo California.

Kolen W.J. and Kroon L.G., (1991), On the computational complexity of (maximum) class scheduling, *European Journal of Operations Research* 54, pp. 23-38.

Kramer F.J. and Lee C.Y. (1994), Due window scheduling for parallel machines. *Mathematical and Computer Modelling*, Volume 20, Issue 2, , 69-89.

Kroon L.G., Salomon M. and Van Wassenhove L. N. (1995), Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research*, 82, 190-205.

Laguna M. and Gonzalez-Velarde, (1991), A search heuristic for just-in-time scheduling in parallel machines, *Journal of Intelligent Manufacturing* 2, pp. 253-260.

Lann A. and Mosheiov G. (2002), A note on the maximum number of on time jobs on parallel identical machines. *Computers and Operations Research* (Article in press)

Larrañeta, J.L.(1977), Programación Lineal y Grafos, Publicaciones de la Universidad de Sevilla, Serie Ingeniería, nº1, 296-305.

Lawler E.L., Lenstra, J.K., Rinnooy Kan A.H.G. and Shmoys, D.B. (1993), Sequencing and Scheduling: Algorthms and complexity. In Logistic of Production and Inventory, Handbooks in Operations Research and Management Science 4, North-Holland.

Li S. (1997), A hybrid two-stage flowshop with part family, batch production, major and minor set-ups. *European Journal of operational Research* 102, 142-156.

Linn R., Zhang W. (1999), Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering* 37, 57-61.

Nowicki E., Smutnicki C. (1998), The flow shop with parallel machines: A tabu search approach. *European Journal of Operations Research* 106, 226-253.

Paul R.J. (1979), A production scheduling in the glass container industry. *Operations Research* 22, 290-302.

Pinedo M. (1995), Scheduling. Theory, Algorithms, and Systems. Prentice Hall international series in industrial and systems engineering. ISBN 0-13-706757-7.

Pitsoulis L. S. and Resende M.G.C. (2002), Greedy randomized adaptive search procedures, *Handbook of Applied Optimization*, P.M. Pardalos and M.G.C. Resende, Eds., Oxford University Press, pp. 168-183.

Ramudhin, A. and Ratliff, H. D. (1992), Generating Daily Production Schedules in Process Industries, *Working paper* 92-51, Faculté des Sciences de L'Administration, Université Laval, Québec, Canada.

Ramudhin A. and Ratliff H.D. (1995), Generating daily production schedules in process industries. *IIE Transactions*, 27, 646-656.

Resende M.G.C. (1998), Computing Aproximate Solutions of the Maximum Covering Problem with GRASP. *Journal of Heuristics*, 4, 161-177.

Resende M. G. C. and Ribeiro C. C. (2003), Greedy randomized adaptive search procedures, in Handbook of Metaheuristics, F. Glover and G. Kochenberger, eds., Kluwer Academic Publishers, pp. 219-249.

Riane F., Artiba A. and Iassinovski S. (2001), An integrated production planning and scheduling sustem for hybrid flowshop organizations. *Int. J. Production Economics* 74, 33-48.

Rinnooy Kan A.H.G. (1976), Machine Scheduling Problems: Classification, Complexity and Computations, Nijhoff, The Hague.

Salvador M.S. (1973), A solution to a special case of flow shop scheduling problem, in: Elmaghraby, S.E. (Ed.). *Symposium of the theory of scheduling and applications*, Springer, New York.

Sivrikaya-Serifoglu F. and Ulusoy G. (1999), Parallel machine scheduling with earliness and tardiness penalties. *Computers and operations Research* 26, 773-787.

Sriskandarajah C. (1993), Performance of scheduling algorithms for no-wait flowshops with parallel machines. *European Journal of Operations Research*, 70, 365-378.

Sundararaghavan P. and Ahmed M. (1984), Minimizing the sum of absolute lateness in single machine and multimachine scheduling. *Naval Research Logistic*, 31, 325-333

Zhu Z. and Heady R.B. (2000), Minimizing the sum of earliness/tardiness in multi-machine scheduling: a mixed integer programming approach. *Computers and Industrial Engineering* 38, 297-305.

Haouari M. and M'Hallah R. (1997), Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters* 21, 43-53.

Oguz C., Zinder Y., Ha Do V., Janiak A. and Lichtenstein M. (2003), Hybrid flow-shop scheduling problems with multiprocessor task systems. *European Journal of Operational Research*. In Press.

Wardono B. and Fathi Y. (2003), A tabu search algorithm for the multi-stage parallel machine problem with limited buffer capacities. *European Journal of Operational Research*. In Press.

Lee I. (2001), Artificial intelligence search methods for multi-machine two-stage scheduling with due date penalty, inventory, and machining costs. *Computers & Operations Research* 28, 835-852.

Tardos E. (1985), A strongly polynomial minimum cost circulation algorithm. *Combinatorica* 5, 247-255

Glover F. and Klingman (1976), Networks applications in industry and government. *AIIE Trans*. B 9, 363-376.

Esau L.R and Williams K.C. (1966), On teleprocessing system design II. *IBM Systems J.*, B 5, 142-147.

Elmaghraby, S.E. (1978), Activity Networks: Project Planning and Control by Network Models. Wiley-Interscience, New York, N.Y.

Levner, E.V. and Nemirovsky A.S. (1991), A network flow algorithm for just-in-time project scheduling. *Memorandum COSOR 91-21*, Dept. of Mathematics and Computing Science, Eindhoven University of Technology, Eindhoven.

Jewel W.S. (1957), Warehousing and distribution of a seasonal product. *Nav. Res. Logistic* Q. B 4, 29-34.

Freedman M.L. and Tarjan R.E. (1984), Fibonacci heaps and their uses in improved network optimization algorithms. *Pro. 25th IEEE Symp. On the Foundations of Computer Science*.