PROYECTO FINAL DE INGENIERÍA INDUSTRIAL

ALGORITMO PARA LA OPTIMIZACIÓN EN EL PLANEAMIENTO Y LA PROGRAMACIÓN DE LA PRODUCCIÓN

Santiago M. Segarra
47255

Tutor: Lic. Francisco Villaverde

2011

Santiago Martín Segarra.
ALGORITMO PARA LA OPTIMIZACIÓN EN EL PLANEAMIENTO Y LA PROGRAMACIÓN DE LA PRODUCCIÓN.
Buenos Aires, Argentina.
Junio de 2011.
128 p.
TESIS DE GRADO.

AGRADECIMIENTOS

A mi familia, mis amigos y mi novia, por su constante apoyo.

Al Lic. Francisco Villaverde, mi tutor y al Ing. José Jalil, por sus valiosos aportes.

“Logic will get you from A to B. Imagination will take you everywhere.”

“If you can't explain it simply, you don't understand it well enough.”

- Albert Einstein

RESUMEN

La inherente complejidad de la resolución del problema general de planeamiento y programación de la producción atrae a diversas ramas de la matemática y computación aplicada. En el presente trabajo, se desarrolla un análisis de optimización matemática a través de programación lineal entera mixta que busca resolver dicho problema. Además del desarrollo tradicional, se agrega una serie de estrategias que buscan reducir la complejidad del problema a resolver, mejorando la performance del algoritmo. Luego, se explicita su aplicación computacional y se muestran sus resultados para algunos casos de ejemplo. Por último, se detallan sus limitaciones así como el alcance de su aplicación y posibles puntos de mejora.

ABSTRACT

The inherent complexity of the general production planning and scheduling problem attracts several fields of study from applied mathematics and computer science. In this thesis, the problem is approached through a mathematical optimization analysis using linear mixed integer programming. Apart from the traditional development, the tool developed includes a series of strategies conceived to reduce the problem complexity, increasing the algorithm's performance. Furthermore, a possible computational application is suggested and the results from some example cases are shown. Finally, the tool's limitations are detailed as well as some possible investigation directions to overcome them.

ÍNDICE

1 INTRODUCCIÓN

El planeamiento y la programación de la producción son actividades esenciales para cualquier empresa productiva sin importar su industria, su origen o su locación geográfica. Son las herramientas que la empresa posee para ajustar las actividades dentro de sus fábricas tanto con la estrategia corporativa como con la demanda del mercado. Por consiguiente, excepto en empresas muy pequeñas donde el costo de obtener datos y procesarlos puede superar el beneficio de la planificación productiva, su impacto en los beneficios obtenidos suele ser significativo.

La dificultad de generar planes y programas de producción adecuados depende de una gran cantidad de factores. Entre éstos se pueden incluir la complejidad del proceso productivo, la cantidad de productos finales e intermedios, la predictibilidad de la demanda y los recursos y el horizonte de estudio. Para los casos más sencillos, basta con análisis aislados y la experiencia de un grupo de empleados para generar planes y programas que maximicen el beneficio. Sin embargo, para problemas de mayor complejidad, se requieren herramientas más poderosas para diseñar planes y programas óptimos. En el presente trabajo, se presenta una posible herramienta que aplica conceptos de optimización matemática lineal con el objetivo de resolver un problema universal de planeamiento y programación de la producción.

En este punto es necesario introducir el criterio de optimalidad que regirá a lo largo de todo el presente trabajo. Un plan y un programa de producción se denominarán óptimos cuando sean aquellos que, teniendo en cuenta las características del mercado, mejor se alineen con la estrategia corporativa. En este caso en particular, se fija el objetivo corporativo de maximizar el beneficio económico obtenido durante el horizonte de estudio. De esta forma, se puede definir una 'distancia al óptimo' basada en la diferencia en beneficio obtenido por el plan (o programa) estudiado en comparación con el óptimo.

Antes de continuar, resulta necesario introducir formalmente los conceptos de planeamiento y programación de la producción.

1.1 Definiciones de planeamiento y programación de la producción

Resulta frecuente la confusión entre estos dos términos. Sin embargo, estando lejos de ser sinónimos, comprender la diferencia entre ellos es esencial para entender el desarrollo del presente trabajo.

Las definiciones convencionales de planeamiento y programación desde un punto de vista general son las siguientes:

- *Planeamiento:* Modelo sistemático de una actuación pública o privada, que se elabora anticipadamente para dirigirla y encauzarla [Real Academia Española, 2011]. Como datos de entrada se requieren una descripción del estado inicial, una especificación del estado objetivo y una enumeración de las posibles actividades a desarrollar. La salida, es decir la secuencia de actividades que lleva del estado inicial al objetivo, se denomina plan [Pool *et al.*, 1998].
- *Programación:* Asignación exacta de recursos a actividades respetando restricciones de duración, precedencia, capacidad e incompatibilidad [Brusoni *et al.*, 1996]. El conjunto de actividades, la serie de recursos y la especificación de todas las restricciones son la información de entrada. Mientras que la asignación de recursos, llamada programa, constituye la salida.

En el ámbito industrial la distinción es más sutil. El planeamiento de la producción continúa estando más relacionado con la determinación de actividades mientras que la programación de la producción se ocupa de la asignación de los recursos. Sin embargo, desde este punto de vista los límites se encuentran algo difusos. Por esta razón, en el ámbito productivo, la mayor diferencia entre ambos conceptos reside en la resolución y el horizonte de estudio. El planeamiento posee una menor resolución (sectores productivos y semanas) y un mayor horizonte (algunos meses), mientras que la programación presenta una mayor resolución (máquinas y horas) y un menor horizonte temporal (semanas).

De esta forma, el ***plan de producción*** se vuelve un documento con mayor nivel de agregación, de naturaleza táctica dentro de la empresa. Su concepción suele realizarse en niveles gerenciales superiores a los del ***programa de producción***, no siendo el detalle de las restricciones lo esencial sino la definición de los grandes aspectos a seguir. El ***plan*** actúa entonces como información de entrada para el ***programa***, documento éste de mayor resolución, menor nivel jerárquico y

de naturaleza netamente operativa. De esta forma, el programador intenta asignar los recursos disponibles para cumplir con los objetivos impuestos en el **plan**. En otras palabras, el **plan** enuncia lo que se producirá con cierto nivel de agregación mientras que el **programa** detalla lo que cada máquina producirá en cada instante para lograr cumplir con el primero.

La figura 1.1 esquematiza la relación ideal entre ambos procesos. La información de entrada tanto del planeamiento como de la programación de la producción no pretende ser exhaustiva, sino que se explicitan algunos puntos fundamentales.

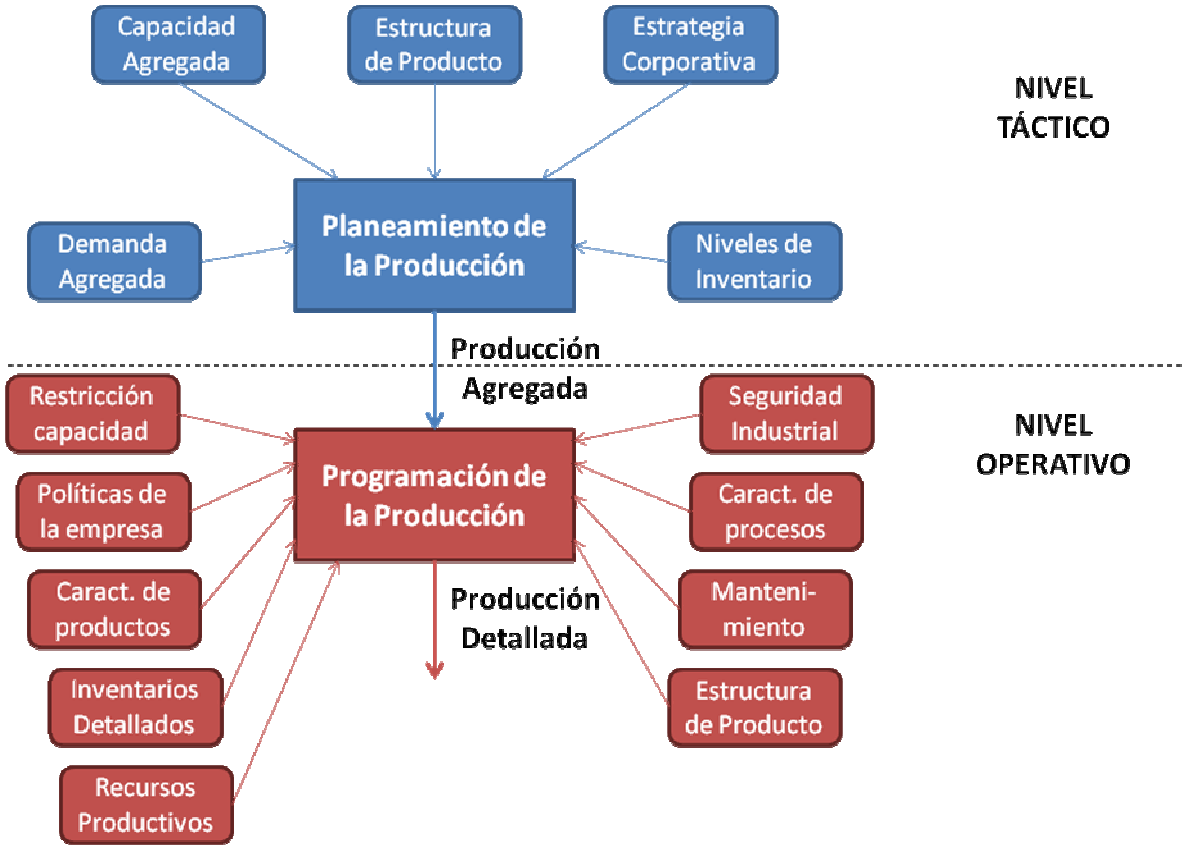

Figura 1.1: Relación ideal entre planeamiento y programación de la producción

Como se puede observar, existe una íntima relación entre el plan y el programa de producción, cuya dinámica vale la pena analizar.

1.2 Relación entre el plan y el programa de producción

Como se explicó, es frecuente la situación en que estos dos procesos son desarrollados por separado en distintas áreas de la empresa de manera secuencial donde el primero actúa como dato de entrada del segundo. Esta

separación jerárquica aparece como la solución natural a la creciente complejidad del problema. Es decir, atacar el problema del planeamiento con el nivel de resolución de la programación conllevaría el manejo de un enorme número de variables y restricciones incrementando así la complejidad de su resolución. Al dividirlo de manera jerárquica, se resuelven dos problemas separados, cuya complejidad individual es mucho menor.

Sin embargo, la resolución por separado conlleva algunas complicaciones adicionales. Así se presenta el caso de una solución que aparenta ser óptima al nivel de planeamiento y puede ser de dificultosa implementación al nivel del programa. Más aún, es común que el plan óptimo no sea factible a nivel de programación. Esto puede ser producido por una serie de razones. En primer lugar, el hecho que la dinámica de la producción respete las restricciones agregadas de capacidad y materia prima no asegura que respete las detalladas. Además, los procesos complejos poseen detalles que no se tienen en cuenta al nivel del planeamiento como *setups* (puesta a punto de la máquina cuando se va a comenzar un lote de producción), recalentamientos (en procesos a altas temperaturas), manejo de productos secundarios y capacidad de almacenes intermedios. Además, al realizar el plan y el programa por separado, suele surgir en algunas empresas una incompatibilidad de naturaleza aún más profunda. En muchos casos el objetivo (y el criterio de evaluación) de ambos procesos es distinto. El planeamiento busca satisfacer la demanda reduciendo inventarios y ventas no atendidas mientras que el programa busca mejorar medidas operativas de producción como el nivel de utilización de las máquinas o el tiempo promedio de fabricación. Esta claro que en una empresa ideal, donde la jerarquización se realiza de manera perfecta bajo un objetivo común, este último problema no debería existir. Sin embargo, es un problema recurrente en la actividad cotidiana de muchas empresas.

En consecuencia, no siempre el programa puede cumplir con el plan. Esto genera un retroceso de la información (*backtracking*) del programador al planificador explicando las causas de la imposibilidad de la aplicación del plan. El *backtracking* conlleva una serie de ineficiencias y pérdidas de tiempo para todo el sistema. Sin embargo, si se busca aumentar el nivel de información a la hora de planificar para reducir las ineficiencias por *backtracking,* se aumenta la complejidad del planeamiento perdiendo la ventaja del análisis por separado.

Por consiguiente, se puede observar que existe un delicado compromiso entre la complejidad a manejar y la correcta correlación entre el plan y el programa de producción para reducir el *backtracking*.

2 OBTENCIÓN DEL PLAN Y PROGRAMA DE PRODUCCIÓN

En este capítulo se realizará una breve reseña de la evolución de las herramientas tradicionales para la obtención del plan y el programa de producción. Además, a modo de ejemplo, se mencionarán algunos desarrollos novedosos en el área. Por último, se explicitarán las características principales de la herramienta a desarrollar en los siguientes capítulos del trabajo.

2.1 Herramientas tradicionales

En la presente sección, se repasarán algunas herramientas tradicionales para resolver, de manera parcial o total, los planes y programas de producción. La lista no intenta ser una enumeración exhaustiva sino que el objetivo es mostrar la evolución de los métodos de la mano de una mayor aplicación tecnológica. Tampoco es el objetivo del presente trabajo un análisis profundo de cada herramienta sino un repaso superficial de sus características distintivas.

- *Control Estadístico de Inventario (SIC)*
  De fácil automatización, el SIC utiliza información histórica y pronósticos para controlar los niveles de inventario con herramientas matemáticas. Su utilización tiene buenos resultados en ambientes más bien estáticos ya que los pronósticos basados en información histórica pierden validez con la creciente complejidad. Además, la aplicación aislada del SIC en distintas etapas de la cadena productiva genera el llamado efecto “Forrester” [Forrester, 1958] donde decisiones independientes generan niveles de inventarios elevados y poco balanceados a lo largo de la cadena.

- *Planificación de Requerimientos de Material (MRP I)*
  Luego de finalizada la segunda guerra mundial, la tecnología computacional que se desarrolló para el manejo de la logística bélica migró hacia el ámbito empresarial. De esta manera, con la introducción de las primeras computadoras en las empresas, la capacidad de cálculo y de manejo de información creció notablemente. Con este crecimiento, en la década de 1950, surge el desarrollo de MRP I. Es un sistema conceptualmente sencillo que genera un plan de producción a partir de la demanda de productos terminados (demanda independiente) que se asume conocida, la estructura

de los productos (a través de la lista de materiales) y los niveles iniciales de inventario. A partir de estos datos se generan, hacia atrás temporalmente, las órdenes de producción (demanda dependiente) de productos semi elaborados, desarrollando un plan de producción que satisface la demanda original optimizando los inventarios intermedios.
En otras palabras, a partir de un plan de producción de productos terminados, con MRP I se calcula qué componentes se requieren, en qué cantidades y en qué momento. Sin embargo, no se tienen en cuenta limitaciones de capacidad productiva.

- *Planeación de los Recursos de Fabricación (MRP II)*
  El MRP II surge como una ampliación del MRP I ya que incluye el cálculo de la capacidad productiva necesaria (antes supuesta ilimitada) y además considera aspectos financieros. Mientras que el MRP I permite la coordinación de la compra de materias primas, el MRP II facilita el desarrollo de un programa de producción que tiene en cuenta la capacidad de las máquinas y de los empleados. Además, transmite información sobre costos a los sistemas de contabilidad y finanzas [Monk & Wagner, 2006].

- *Planificación de Recursos Empresariales (ERP)*
  El término ERP se utilizó por primera vez en 1990 como una extensión del MRP I y MRP II en conjunción con el CIM (Computer-Integrated Manufacturing). Este último término indica la utilización de computadoras para el control completo de la producción. ERP está definido como una arquitectura de software que facilita el flujo de información entre las distintas áreas de la empresa como la productiva, la logística, las finanzas y los recursos humanos [Hicks, 1997].
  ERP provee un eje común a toda la organización permitiendo la estandarización de los sistemas de información internos. Entre sus tareas se incluye tanto el seguimiento de los almacenes como la administración de los recursos humanos, por nombrar dos ejemplos. En otras palabras, los sistemas ERP proveen con la información adecuada, a la gente adecuada en el momento adecuado [Sheridan, 1995]. Esto permite la integración de lo que, en el mejor de los casos, serían centros automatizados por separado.

- *Planificación y Programación Avanzada (APS)*
  APS intenta llevar el concepto del ERP al extremo, incluyendo información externa a la empresa e integrándola en la cadena de información total. Además, aprovecha la mayor capacidad de procesamiento actual para

actualizar sus resultados en tiempo real. Por ejemplo, adapta los planes de producción a medida que se recoge nueva información más exacta del mercado. Por consiguiente, resulta una herramienta de mayor utilidad para la aplicación en ambientes de dinámicas complejas.

Hasta aquí se intentó presentar en forma cronológica y, por consiguiente, de complejidad creciente, algunas de las herramientas tradicionales para resolver los problemas de planeamiento y programación de la producción.

Si bien muchas de estas herramientas resultan de gran utilidad para la integración de toda la empresa, tanto MRP I como MRP II y los módulos de planeamiento del ERP y APS son una serie de heurísticas que buscan un plan y programa de producción adecuado. Sin embargo, algunos autores argumentan que se pueden encontrar mejores resultados aplicando modelos de programación matemática más poderosos como los de programación entera mixta [Wolsey, 2006].

Por lo tanto, en la próxima sección se presentan, a modo de ejemplo, algunos de los desarrollos más recientes en las áreas de planeamiento y programación de la producción. El objetivo de dicha sección no es enumerar y menos explicar el estado actual de esta área del conocimiento, sino demostrar que su complejidad hace que se continúen realizando constantes investigaciones y mejoras.

2.2 Algunos desarrollos novedosos

La gran aplicabilidad y la complejidad desafiante de los problemas de planeamiento y programación de la producción atraen la atención de muchas áreas de la ciencia de la computación. Como consecuencia se puede apreciar un extenso número de variantes novedosas que aportan su perspectiva a la resolución del problema general. A modo de ejemplo, a continuación se comentarán algunas de ellas.

Comenzando por la más tradicional aplicación de la ***programación lineal*** al problema general del planeamiento de la producción [Hackman & Leachman, 1989] se puede continuar por una mejora posterior que combina elementos de programación lineal con parámetros variables en el tiempo y utiliza ***simulación*** como la herramienta para evaluar la performance del plan de producción [Hung & Leachman, 1996]. Otros autores, detectan los '*cuellos de botella*' y optimizan su utilización, construyendo el resto del programa de producción a partir de allí [Lee & Kim, 2002].

También se ha aplicado la ***teoría de colas*** a la resolución del problema. Si bien esta teoría generalmente aporta información sobre los 'estados permanentes' y no

sobre los 'estados transitorios' de los sistemas, su combinación con un ***sistema hidráulico*** (donde los materiales se representan como fluidos y las máquinas como válvulas) ha sido capaz de aportar información valiosa para períodos de estudio de corta duración [Dai & Weiss, 2002].

Por último, existe toda otra serie de autores que busca mejorar la relación entre el planeamiento y la programación de la producción para evitar ineficiencias y planes no factibles. Por ejemplo, algunos autores han generado un módulo intermedio de resolución que funciona como amalgama entre ambos: una 'programación de alto nivel' [Cai et al., 2011].

La vasta literatura sobre el tema es un claro indicio de su importancia y vigencia en la vanguardia de la ciencia aplicada a la ingeniería industrial.

2.3 Enfoque de la herramienta a desarrollar

Dentro de este marco es más sencillo presentar el enfoque de la herramienta que se desarrollará a lo largo del trabajo. Esta herramienta se desarrolló para un concurso en donde la factibilidad del programa de producción era de vital importancia. Es decir, si se generaba un programa no factible, entonces la solución carecía de valor ante el jurado de dicho concurso. Por esta razón, se decidió priorizar la mencionada factibilidad.

Es sabido que gran parte de los problemas de factibilidad del programa se deben a que éste debe responder a un plan en dónde las restricciones agregadas no reflejan del todo la realidad de la planta. Por esta razón, en el presente trabajo se toma otro enfoque: se utiliza programación entera mixta para diseñar el programa sin pasar por el plan de producción. Lógicamente, el plan (con el mismo horizonte que el programa diseñado) queda confeccionado ya que el paso desde el programa al plan no requiere de ningún trabajo al ir de mayor a menor resolución. De esta forma, se asegura la factibilidad del programa de producción.

Sin embargo, este enfoque trae una obvia complicación que es el aumento de la complejidad en el cálculo. Al encarar todo el horizonte de estudio desde la resolución del programa, la optimización se vuelve un proceso de alto consumo computacional. Por consiguiente, se desarrollan distintas estrategias para aumentar la performance del algoritmo total.

El desarrollo de los algoritmos de base así como de las estrategias mencionadas para reducir el tiempo de procesamiento se puede observar en las siguientes secciones. Además se presentan, a modo de ejemplo, aplicaciones de la herramienta desarrollada.

3 PROBLEMA ACOTADO DE PROGRAMACIÓN DE LA PRODUCCIÓN

3.1 Introducción

Como metodología de aproximación a la solución del problema general de programación de la producción, se comenzará analizando un modelo acotado para luego ir agregando complejidad que acerque el modelo a la realidad.

Definir el alcance del modelo acotado pareciera ser una tarea algo subjetiva, ya que no queda del todo claro en qué momento un modelo pasa de ser "acotado" a "completo". Sin embargo, se intentará librarse de la subjetividad tomando el siguiente criterio: "Un modelo acotado es aquel que puede representarse en programación continua, es decir, prescindiendo de variables de decisión enteras y binarias."

Este criterio no es adoptado de forma caprichosa o al azar. Por el contrario, tiene una importante influencia en el método matemático que puede utilizarse para su resolución. En los problemas continuos, pueden utilizarse algoritmos del tipo Simplex (Primal, Dual, etc.) mientras que en problemas enteros mixtos (aquellos que combinan variables enteras con continuas) se deben utilizar algoritmos de corte de espacios factibles como el Branch & Bound. Estos algoritmos suelen tener una performance notablemente menor a la hora de resolver problemas con altos volúmenes de información. De todas formas, esta disquisición se ampliará más adelante cuando se hayan mencionado los conceptos necesarios para su análisis.

3.2 Descripción del problema

En esta sección, se intentará dar un marco formal al "Problema Acotado de Programación de la Producción" (PAPP). Es decir, se describirá la información de entrada, las restricciones a considerar, el funcional y la información de salida a obtener.

En el trabajo cotidiano, el problema de programación de producción es algo que se suele describir de manera poco estructurada entre empleados, asumiendo que todos tienen un nivel de conocimiento necesario para completar información inconclusa o ausente. Por ejemplo, resultaría impensable que en una reunión entre el director de producción y el jefe de planta en la que se va a discutir este tema comiencen aclarando qué es una máquina o qué máquinas están disponibles. Sin embargo, el modelo matemático, una abstracción del problema

real, debe ser correcto y estructurado. “Correcto” en el sentido de que las soluciones e inferencias que se realicen a partir del modelo deben tener un nivel de detalle alineado con los requerimientos de factibilidad (a través de las restricciones) y optimalidad (a través del funcional). “Estructurado” en el sentido de que debe utilizar objetos y restricciones estándar ya que debe ser un modelo lo suficientemente general como para aplicarse tanto a una fábrica de medicamentos como a una productora de aeronaves, por ejemplo.

Se comenzará describiendo la estructura generalizada del proceso de producción. A continuación, la figura 3.1 muestra un esquema que facilitará su comprensión:

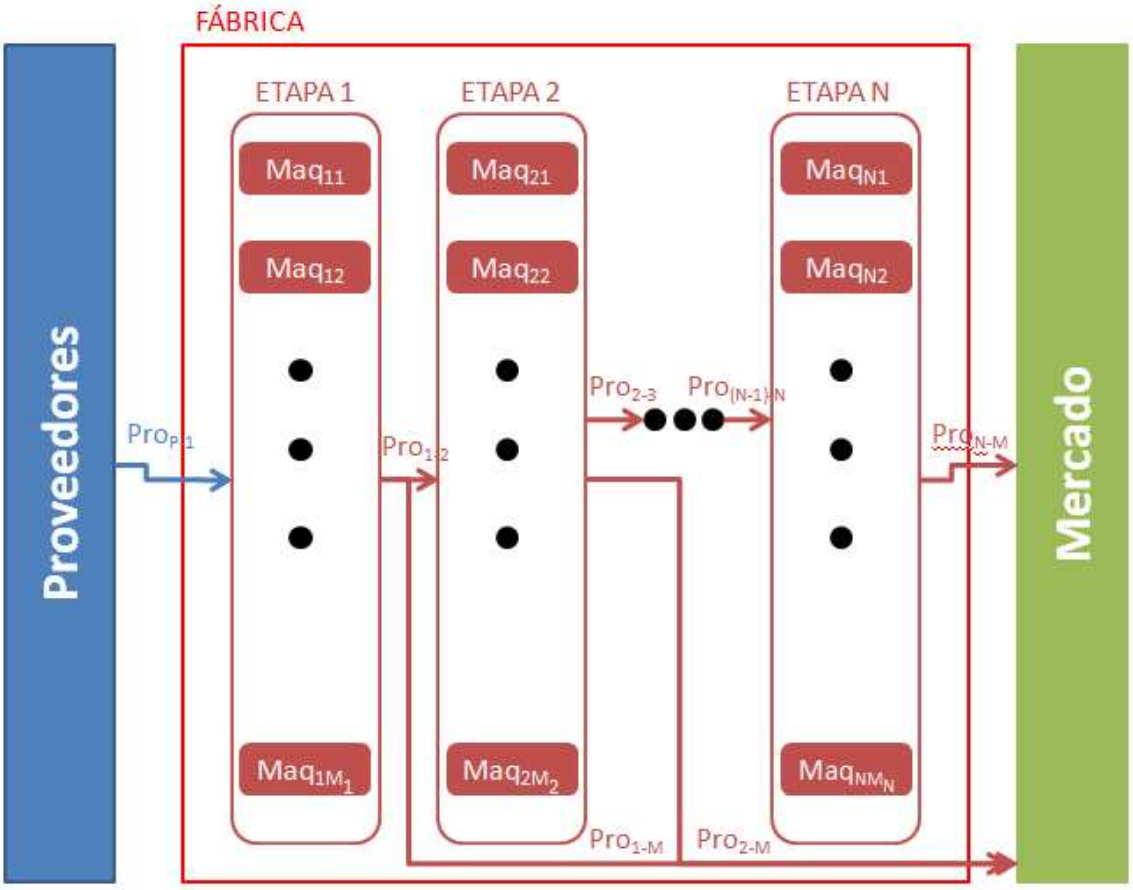

Figura 3.1: Esquema de un proceso generalizado de producción

A continuación se explicarán todos los componentes del esquema y se detallará su alcance dentro del PAPP:

- *Proveedores:* Su función es proveer al proceso productivo con materias primas e insumos productivos “externos” necesarios para su correcto funcionamiento. En el esquema, por simplicidad se describió únicamente su

relación con la primera etapa de producción. Sin embargo, en una situación generalizada los proveedores tienen relación con las N etapas productivas. De todas formas, NO se incluirá el estudio de proveedores dentro del marco del PAPP. Es decir, se considerará a la materia prima e insumos "externos" como de costo nulo y disponibilidad infinita. En otras palabras, no se tendrán en cuenta estrategias como desarrollo de proveedores, tipo de aprovisionamiento y descuentos por volumen a la hora de optimizar el programa de producción.

- *Mercado:* Es aquella entidad abstracta que demanda los productos que se venderán. No todos los productos son destinados al mercado, sino que pueden ser destinados a stock de semi elaborados para continuar su proceso productivo. La demanda de cada producto es determinada y conocida a la hora de optimizar el PAPP.

- *Fábrica:* Es la entidad sobre la cual se realizarán las decisiones de producción para lograr optimizar el PAPP. La fábrica consta de N etapas productivas y la etapa "i" cuenta con M_i máquinas.

- *Etapa productiva:* Se llamará de este modo a cualquier proceso o conjunto de éstos necesarios para seguir el proceso productivo general. Por ejemplo, si se estudia una fábrica productora de elásticos teñidos, se podrían diferenciar 4 etapas productivas: "urdido", "confección", "teñido" y "terminado". Cada una de ellas requiere que la etapa anterior haya tenido lugar para poder llevarse a cabo. Sin embargo, no todas las etapas productivas tienen que haber ocurrido para que el producto esté preparado para ser demandado en el mercado. Es decir, los productos de cada etapa tienen dos destinos: la etapa siguiente (como semi elaborado) o el mercado. En el ejemplo antes mencionado, la fábrica puede vender el producto final o puede vender el elástico crudo (es decir, sin teñir) a otra fábrica, por ejemplo. Una implicación directa es que los productos de la última etapa productiva (N) deben ser producidos exclusivamente para su venta en el mercado.

- *Máquina:* La definición de "máquina" dentro de este problema es un tanto más amplia que la definición literal del concepto. Es decir, a los efectos del problema, se considerará "máquina" a cualquier entidad "i" que por diseño puede producir una cantidad k_i de productos ($k_i>0$) a un precio por unidad y a un ritmo determinado. Desde este punto de vista, se puede llamar

"máquina" tanto a la entidad definida literalmente, a un grupo de trabajo humano o a alguna combinación intermedia de estos. Sin embargo, es lógico pensar que la validez del determinismo en el costo y el ritmo de producción decrece con el aumento del componente humano dentro de la entidad "máquina" del proceso productivo. Cabe aclarar que las M_j máquinas pertenecientes a la etapa productiva "j" son, en general, distinguibles entre ellas. Esta distinción existe a dos niveles. En primer lugar, los k productos que cada máquina puede producir (por diseño) pueden ser distintos. En segundo lugar, aún fabricando los mismos productos las máquinas pueden diferir en su costo por unidad y su ritmo de producción.

- *Producto:* Existen dos grandes categorías dentro de los productos: terminados y semi elaborados. Se llamarán "terminados" a aquellos productos que se comercializan en el Mercado y "semi elaborados" a aquellos que se utilizan como materia prima para producir otros productos. Haciendo referencia al esquema anterior, se llamará $Pro_{i-(i+1)}$ al conjunto de productos semi elaborados fabricados en la etapa "i" para ser utilizados en la etapa "i+1". A su vez, se llamará Pro_{i-M} al conjunto de productos terminados fabricados en la etapa "i" para ser vendidos en el mercado. Es importante notar que ambas definiciones no son excluyentes, es decir, puede haber productos que sean tanto "terminados" como "semi elaborados" simultáneamente. En otras palabras, dado una etapa "i", un mismo producto puede estar dentro de $Pro_{i-(i+1)}$ y Pro_{i-M}. Para dejar en claro la función de los productos semi elaborados, es esencial comprender que todo producto fabricado en una etapa distinta de la primera, requiere de cierta cantidad de materia prima semi elaborada producida por la etapa inmediatamente anterior. En el ejemplo de la fabricación de elásticos, si el producto que se comercializa es un "carretel de elástico teñido", el semi elaborado necesario para la etapa de terminación es cierta cantidad de "elástico teñido" por cada producto final (el carretel entraría dentro de los proveedores de insumos, pero se aclaró que no se tendrían en cuenta en el estudio). Para producir el "elástico teñido", se necesita el "elástico crudo" que se genera en la etapa de confección y así sucesivamente. Escribiendo esto formalmente, si se denomina SE(i) al conjunto de los semi elaborados necesarios para producir el producto "i":

$$Si\ y \in SE(x)\ con\ x \in \left(Pro_{i-(i+1)} \bigcup Pro_{i-M}\right),\ i > 1 \rightarrow y \in Pro_{(i-1)-i}$$

Esencialmente, lo que se explicita aquí es que si el producto “y” es un semi elaborado necesario para producir un producto “x” de la etapa “i”, entonces “y” pertenece a $Pro_{(i-1)-i}$. Es importante notar que en esta definición nada se dice sobre la cantidad necesaria del producto semi elaborado “y” para producir una unidad de “x”. Este tema se aclarará más adelante.

Ahora que se comprende la estructura del proceso productivo general, se pasará a detallar la información de entrada necesaria para la resolución del PAPP.

Antes de comenzar cabe aclarar lo siguiente:

i) *Unidad temporal:* Parte de la información depende de una unidad temporal. Por ejemplo, ritmos de producción o demanda se pueden definir como unidades físicas sobre unidad temporal. Esta unidad temporal depende de la naturaleza del problema y puede ir desde minutos hasta años. Para independizarnos de dicha variabilidad, se tomará la unidad [utem] como unidad temporal estándar, que luego se podrá interpretar de distintas formas para ajustar el problema a distintas situaciones.

ii) *Unidad física:* Parte de la información depende de unidades físicas de producto. Por ejemplo, la cantidad de semi elaborado necesaria para realizar una unidad de producto final. Sin embargo, la unidad utilizada depende de la naturaleza de los productos, podría ser litros, kilos, metros y demás. Para evitar este problema, se utilizará una unidad estándar a todos los productos que se denominará [ust].

Una vez aclarado esto, se pasará a definir dos conjuntos a partir de las referencias de la Figura 3.1. Se llamará “$Tot_{Productos}$” al conjunto de todos los productos terminados y semi elaborados de la fábrica. Análogamente, se llamará “$Tot_{Maquinas}$” al conjunto de todas las máquinas dentro de la fábrica. Es decir:

$$Tot_{Productos} = \bigcup\left(\left(\bigcup_{i=1}^{N-1} Pro_{i-(i+1)}\right),\left(\bigcup_{i=1}^{N} Pro_{i-M}\right)\right)$$

$$Tot_{Maquinas} = \bigcup_{i=1}^{N}\bigcup_{j=1}^{M_i} Maq_{ij}$$

Sobre estos dos conjuntos, se definirán las funciones de entrada necesarias. Por lo tanto, en primer lugar se necesita la información suficiente como para definir

estos dos conjuntos. A continuación se detalla, por categorías, toda la información de entrada necesaria para definir inequívocamente un PAPP:

- *Productos:* Un listado con el nombre o código de cada producto terminado y semi elaborado.
- *Máquinas:* El número total de etapas productivas (N) y el número de máquinas que posee cada etapa (M_i).
- *Horizonte temporal:* Es un parámetro que indica la cantidad de 'utems' que durará el horizonte de estudio.
- *Lista de Materiales (BOM):* Es una función de dos variables BOM(x,y) que indica qué cantidad de material 'x' hace falta para producir una unidad de material 'y'.
- *Demanda:* Es una función de una variable DEMANDA(x) indica la demanda del material 'x' por unidad de tiempo.
- *Capacidad:* Es una función de una variable CAPACIDAD(x) indica la máxima cantidad de 'utems' que puede trabajar la máquina 'x' en el horizonte temporal.
- *Definición de Materiales:* Es una función vectorial de una variable MATERIALDEF(x) que devuelve el precio de venta, el costo de almacenamiento y el costo de ruptura del material 'x'. (Los costos se explicarán más adelante)
- *Producción de Materiales:* La función de dos variables MATERIALPROD(x,y) es una función vectorial que devuelve el ritmo de producción del producto 'y' en la máquina 'x' y el costo de producir una 'ust' del producto 'y' en la máquina 'x'.

A continuación se presenta la tabla 3.1 resumiendo la información mencionada para facilitar su comprensión esquemática:

Categoría	Nombre	Naturaleza	Dominio	Imagen	Unidad	Tipo de dato
Productos	Productos	Conjunto	-	-	-	String
Máquinas	Total Etapas	Parámetro	-	-	[etapas]	Entero
	N° Máquinas	Función	(i en Total Etapas)	Máquinas en etapa i	[máquinas]	Entero
Horizonte Temporal	Horizonte	Parámetro	-	-	[utem]	Entero
Lista de Materiales	BOM	Función	(x en Tot_Productos, y en Tot_Productos)	Cantidad de x para producir una unidad de y	[ust]	Float
Demanda	DEMANDA	Función	(x en Tot_Productos)	Demanda de x	[ust/utem]	Float
Capacidad	CAPACIDAD	Función	(x en Tot_Máquinas)	Capacidad temporal de x	[utem]	Float
Definición de Materiales	MATERIALDEF	Función	(x en Tot_Productos)	1) Precio de venta de x	[$/ust]	Float
				2) Costo de almacenamiento de x	[$/(ust*utem)]	Float
				3) Costo de ruptura de x	[$/(ust*utem)]	Float
Producción de Materiales	MATERIALPROD	Función	(x en Tot_Máquinas, y en Tot_Productos)	1) Ritmo de producción	[ust/utem]	Float
				2) Costo de Producción	[$/ust]	Float

Tabla 3.1: Resumen de la información de entrada para la definición de un PAPP

Notar que algunos datos de entrada son enteros. Sin embargo, esto no contradice la definición inicial adoptada para el PAPP, ya que las variables que no deben ser enteras para poder utilizar el método simplex son las de decisión. Las variables de entrada podrían ser enteras, como en este caso.

Para hacer referencia a una componente de las funciones vectoriales, se utilizará un subíndice entre el nombre de la función y los argumentos. De esta forma, $MATERIALPROD_1$(“A”, “Maq11”) indica el ritmo de producción del material “A” en la máquina “Maq11” mientras que $MATERIALPROD_2$(“A”, “Maq11”) indica su costo de producción.

Prosiguiendo con la definición del PAPP, se describirá la función objetivo: el beneficio total. Es decir, el objetivo de resolver el PAPP es encontrar el programa de producción que maximice el beneficio total para la compañía. Este último se calcula como facturación total menos costos totales. Notar que se considerará a la facturación como sinónimo de ingreso y a los costos como sinónimos de egresos. Es decir, no se considerarán los aspectos financieros del problema sino estrictamente económicos.

Los costos pueden dividirse en tres tipos: Producción, Almacenamiento y Ruptura.

- *Costo de Producción:* Este es posiblemente el costo más sencillo de comprender. Cada unidad producida tiene un costo asociado (más allá del costo de los semi elaborados). Este costo puede estar compuesto de insumos “externos”, la energía, el prorrateo de la amortización de la

máquina, el sueldo de los supervisores, etc. Para calcular el costo de producción total simplemente hay que sumar el costo de producción unitario de cada unidad producida. Es decir, llamando tot_produ(x,y) a la totalidad de producto 'y' producido en la máquina 'x', entonces:

$$C.de\ Prod. = \sum_{i \in Tot\_Productos} \sum_{j \in Tot\_Maquinas} tot\_produ(j,i) * MATERIALPROD_2(j,i)$$

(1)

- *Costo de Almacenamiento:* Guardar stock tanto de producto semi elaborado como terminado tiene un costo asociado. Este costo está dado por diversos factores que incluyen la inmovilización del capital, la amortización de las estructuras de almacenamiento, el riesgo de obsolescencia, entre otros. En este modelo, todos estos factores se verán representados como un costo de almacenamiento unitario por unidad de tiempo. Es decir, el costo incurrido por almacenar una 'ust' durante una 'utem'. Denominando stock(x,t) a la cantidad de producto 'x' almacenado al momento 't', entonces:

$$C.de\ Almacen. = \sum_{i \in Tot\_Productos} \int_{t=0}^{Horizonte} stock(i,t) * MATERIALDEF_2(i) \qquad (2)$$

- *Costo de Ruptura:* No satisfacer la demanda también involucra sus costos. La ruptura (insatisfacción de la demanda) se puede modelar de distintas maneras: suponiendo venta perdida o venta diferida. En este caso, se supondrá que las ventas se realizan de forma diferida. En otras palabras, la demanda no satisfecha se acumula. De todas formas, existen otros costos asociados a la ruptura que incluyen el debilitamiento de la marca y la pérdida de imagen. Su forma de cálculo es similar al costo anterior. Es decir, llamando ruptura(x,t) a la demanda acumulada por ruptura del material 'x' al momento 't', entonces:

$$C.de\ Ruptura = \sum_{i \in Tot\_Productos} \int_{t=0}^{Horizonte} ruptura(x,t) * MATERIALDEF_3(i) \qquad (3)$$

Con el objetivo de calcular los ingresos, se definirá tot_ventas(x) a la totalidad de las ventas del material 'x', de esta forma:

$$Ingreso = \sum_{i \in Tot\_Productos} tot_{ventas(i)} * MATERIALDEF_1(t) \qquad (4)$$

Finalmente, se puede enunciar el funcional a maximizar:

$$Beneficio\ Total = Ingreso - C.de\ Prod - C.de\ Almac. - C.de\ Ruptura \qquad (5)$$

Continuando con la definición del PAPP, se enuncian las restricciones a tener en cuenta a la hora de resolverlo (la formulación matemática es parte de la resolución del problema):

- *Capacidad de Máquinas:* El tiempo de uso de una máquina dada no debe superar su capacidad.
- *Diseño de Máquinas:* Las máquinas pueden producir únicamente los materiales para los que están diseñadas.
- *Límite de Ventas:* Las ventas no pueden superar el pronóstico más las ventas acumuladas por ruptura.
- *Necesidad de Semi Elaborados:* Para producir una unidad de cualquier producto debe haber en stock los semi elaborados necesarios.

Por último, la información de salida que se quiere obtener como solución del PAPP es la siguiente: “Qué producir en cada máquina para cada instante dentro del horizonte de estudio”.

Notar que justamente la información de salida es la necesaria para realizar la programación de la producción que maximice el beneficio durante el horizonte de estudio.

3.3 Esparcimiento del problema

Antes de pasar a la resolución del PAPP, cabe realizar una observación que será de vital relevancia a la hora de la aplicación de cualquier solución de este problema o similares. Para comprender a qué se refiere el “Esparcimiento del Problema”, se considerará, por ejemplo, el “Ritmo de Producción”. Por lo mencionado hasta este punto, se debe comprender que el ritmo depende tanto del producto a fabricar como de la máquina a utilizar. Es decir, se puede pensar lo siguiente:

$$\text{Ritmo de Producción} = ritmo(m,p)\ con\ p \in Tot_{Productos}, m \in Tot_{Maquinas}$$

En otras palabras, se puede pensar a la función *ritmo* como una función definida sobre un espacio bidimensional discreto formado por $Tot_{Maquinas}$ X $Tot_{Productos}$. Sin embargo, se debe notar que esta función no está definida para cada par (m,p) del espacio mencionado. Es decir, una máquina dada no puede producir cualquier producto dentro de la fábrica. Es más, resulta lógico suponer que la proporción de los pares (m,p) para los cuales esta función se encuentra definida disminuye con la automatización y la especialización de las máquinas. Este es un claro ejemplo de una función esparcida. Una detenida reflexión, revela que el problema a resolver contiene numerosas estructuras esparcidas. Por consiguiente, la explotación de esta característica fundamental del problema es crucial para una aplicación eficiente de cualquier solución propuesta.

La explotación de esta característica consiste en generar estructuras de datos hechas a medida para cada función. Por ejemplo, considerando la función *ritmo*, en lugar de generar un espacio discreto rectangular y definir la función sobre una porción de este espacio, se genera un espacio irregular que contiene únicamente los puntos para los cuales la función está definida. De este modo, se ahorra radicalmente tanto memoria como tiempo de procesamiento.

Para ejemplificar de forma básica la explotación extrema del esparcimiento se supondrá el siguiente problema:

"Se busca variar la magnitud de una fábrica a partir del parámetro de dimensionamiento 'd'. Esta fábrica contiene 'd' etapas productivas, cada etapa con 1 máquina y cada máquina puede producir 1 producto. Lo que se busca es un algoritmo que cargue qué máquinas pueden producir qué materiales y luego (no simultáneamente) lo imprima"

La idea es resolver este problema para distintos valores de 'd', explotando y sin explotar el esparcimiento. Luego, se puede graficar el tiempo de solución o la memoria utilizada en función del parámetro 'd'. Notar que en la figura 3.2, el parámetro 'd' toma valores poco realistas (miles de etapas productivas) para demostrar el comportamiento del algoritmo. De tomar valores más pequeños, el problema sería demasiado sencillo de resolver como para poder detectar variación en los tiempos de resolución.

La evidente trivialidad del problema no evita que sus resultados sean reveladores. En el apéndice A (9.1) se puede observar una posible codificación de la solución

del problema en *IBM ILOG CPLEX*. A continuación, en la figura 3.2 se grafican los resultados obtenidos:

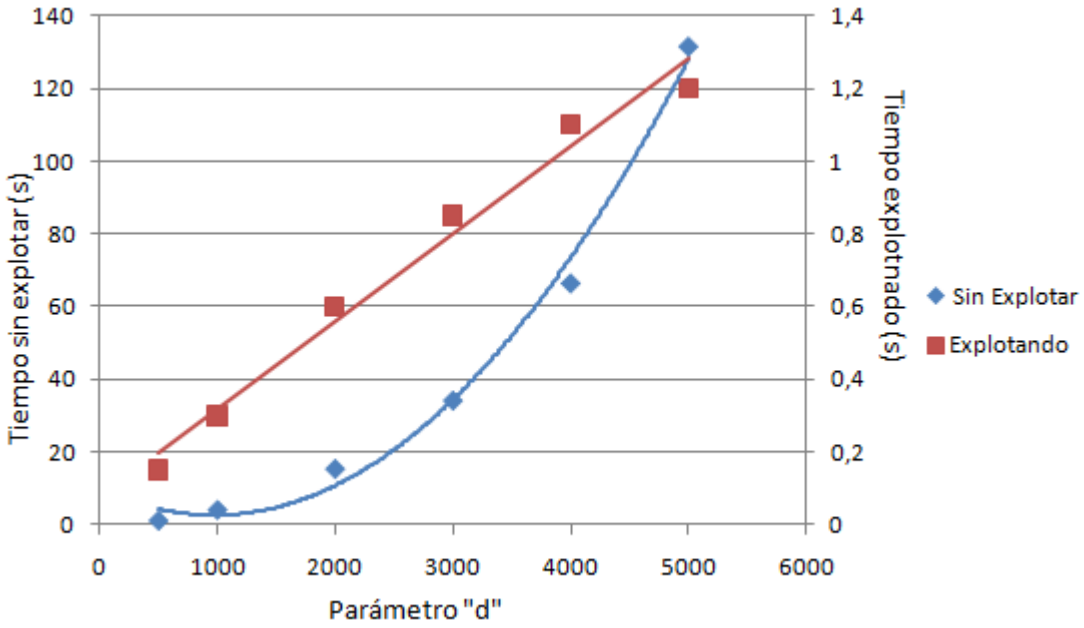

Figura 3.2: Tiempo de resolución para distintas dimensiones del problema explotando y sin explotar el esparcimiento

Más allá de la notable diferencia en los valores absolutos entre los dos métodos para una dimensión dada, lo verdaderamente interesante es ver la evolución de cada curva al incrementarse la dimensión del problema. Como era lógico de esperar, el tiempo de resolución del método “sin explotar” crece de forma cuadrática ya que el espacio de definición depende de d^2. Para el método “explotando”, en cambio, el crecimiento es lineal ya que la combinación (producto, máquina) para los cuales tiene definición el problema, crece proporcionalmente a d.

Análogamente, se puede extrapolar los resultados a espacios α-dimensionales. Es decir, el tiempo necesario para recorrer un espacio α-dimensional sin explotar el esparcimiento es un proceso $O(d^{\alpha})$, mientras que explotando el esparcimiento (en el ejemplo propuesto) puede reducirse hasta O(d).

Cabe mencionar que en casos en los que el esparcimiento no sea tan extremo como en el ejemplo propuesto, la reducción en el orden de crecimiento del tiempo de procesamiento no será tan sencilla de predecir. Sin embargo, para un espacio α-dimensional que presente esparcimiento, el proceso de recorrerlo explotándolo es $O(d^{\beta})$ con $\beta < \alpha$.

3.4 Solución del PAPP

Antes de poder resolver el PAPP, se debe aclarar un concepto delicado relacionado con el tiempo y adoptar una convención adecuada. Muchas de las funciones a utilizar son, en teoría, continuas. Por ejemplo, si se observan las ecuaciones (2) y (3), las funciones *stock* y *ruptura* son idealmente continuas en la dimensión temporal, de allí el cálculo de la integral. Sin embargo, esto implicaría almacenar infinitos valores de la función, lo cual es, obviamente, imposible. La solución a este problema, es discretizar el rango temporal. Es algo similar a lo que se hace en "MatLab" o "Mathematica" para graficar una función continua. Lo que se grafica en realidad es la imagen de un vector de salida con componentes espaciadas un 'paso t' con sus vecinos y luego se unen los puntos de la imagen. Cuando 't' tiende a cero, la función se acerca a su forma continua pero el tiempo de procesamiento tiende a infinito. Por esto, la correcta elección de 't' debe considerar un compromiso entre exactitud y tiempo de procesamiento. En este caso, se tomará como la unidad de discretización la 'utem' por considerarse la unidad de resolución necesaria del programa a desarrollar.

Habiendo aclarado este tema, se pasará a resolver el problema. La resolución será encarada como un proceso de cuatro etapas:

i) Generar nueva información útil a partir de la información de entrada.

ii) Generar la estructura de las variables de decisión.

iii) Expresar el funcional en términos de las variables de entrada y de decisión.

iv) Expresar las restricciones en términos de las variables de entrada y de decisión.

i) Generar nueva información útil a partir de la información de entrada

La idea es generar nuevas variables y conjuntos a partir de la información de entrada para disponer dicha información en un formato que sea de utilidad para resolver el problema.

En este caso, se generarán dos nuevas estructuras de datos:

- *RHorizonte:* Existen muchas funciones que tienen al tiempo como una de sus componentes. Al haber discretizado el tiempo, es necesario un conjunto que guarde sus posibles valores. Es decir, es al análogo temporal de $Tot_{Productos}$ y $Tot_{Maquinas}$.

$$RHorizonte = \{i / \ i \in Z : i > 0 \ \& \ i \leq Horizonte\}$$

- *PosProd:* Con el objetivo de explotar el esparcimiento del problema, se generará un conjunto que indique la "posible producción" (de allí su nombre) para cada 'utem'. Como se sabe que no todas las combinaciones de máquinas y productos están permitidas, se puede asegurar que:

  card(PosProd) < card($Tot_{Maquinas}$)*card($Tot_{Productos}$)*card(RHorizonte)

  De esto se desprende el ahorro en memoria y tiempo de procesamiento. A continuación, la definición formal:

  $$PosProd = \{< i, j, k > / < i, j > \ \varepsilon \ Dom(MATERIALPROD) \ \& \ k \in RHorizonte\}$$

En la tabla 3.2 se resumen las estructuras generadas a partir de la información de entrada.

Nombre	Contenido	Tipo de dato
RHorizonte	Todas las utem desde 1 hasta Horizonte	Entero
PosProd	(i en Tot_Maquinas, j en Tot_Productos, k in RHorizonte)	(String, String, Entero)

Tabla 3.2: Estructuras generadas a partir de la información de entrada en el PAPP

ii) Generar la estructura de las variables de decisión

Las variables de decisión son todas aquellas variables cuyo contenido puede ser determinado y modificado por el algoritmo de resolución. Por ejemplo, "cuántas 'ust' de material "A" producir durante la segunda 'utem'" es una variable de decisión; por el contrario, el precio de venta de una 'ust' del material "A" no es una variable de decisión, ya que no se puede manipular (de hecho, es una variable de entrada).

A continuación se definirán algunas funciones y variables de decisión. Las primeras, al estar definidas en dominios discretos, no son más que conjuntos ordenados de variables de decisión:

- *Stock(i,j):* 'Ust' almacenadas del material 'i' al finalizar la 'utem j'
- *Ventas(i,j):* 'Ust' vendidas del material 'i' durante la 'utem j'
- *Ruptura(i,j):* Demanda acumulada por ruptura en 'ust' del material 'i' al finalizar la 'utem j'
- *Produ(i,j):* 'Ust' producidas del material 'i' durante la 'utem j'

- *Uso(i,j):* 'Ust' utilizadas del material 'i' como semi elaborado durante la 'utem j'
- *MachProd(<i,j,k> ϵ PosProd):* Proporción de la 'utem k' que la máquina 'i' utiliza para producir el material 'j'
- *CAlmac:* Costo Total de Almacenamiento
- *CRuptura:* Costo Total de Ruptura
- *CProdu:* Costo Total de Producción
- *Ingreso:* Ingreso Total por Ventas

En la tabla 3.3 se resume esta información:

Nombre	Dominio	Tipo de dato
Stock	(i en Tot_Productos, j en Rhorizonte)	Float
Ventas	(i en Tot_Productos, j en Rhorizonte)	Float
Ruptura	(i en Tot_Productos, j en Rhorizonte)	Float
Produ	(i en Tot_Productos, j en Rhorizonte)	Float
Uso	(i en Tot_Productos, j en Rhorizonte)	Float
MachProd	<i,j,k> en PosProd	Float
CAlmac	-	Float
CRuptura	-	Float
CProdu	-	Float
Ingreso	-	Float

Tabla 3.3: Resumen de las variables de decisión del PAPP

Notar que todas las variables de decisión son de naturaleza continua, lo que asegura que se podrán usar algoritmos de resolución de la familia del Simplex.

iii) Expresar el funcional en términos de las variables de entrada y de decisión

Este paso es prácticamente trivial luego de haber definido las variables de decisión. Lo que se busca es maximizar el beneficio total. Por lo tanto:

$$Maximizar \qquad Beneficio = Ingreso - CProdu - CAlmac - CRuptura$$

iv) Expresar las restricciones en términos de las variables de entrada y de decisión

A continuación se explicarán y explicitarán todas las restricciones que definen al PAPP. Se buscará otorgarles un nombre distintivo a cada restricción (o grupo de ellas) para futuras referencias.

- *Imagen Acotada:* Si bien todas las variables de decisión pueden tomar valores continuos, no es cierto que las magnitudes físicas representadas por estas variables puedan tomar cualquier valor real. Por lo tanto, es necesario acotar los posibles valores de las variables de decisión:

  R1) $Stock(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  R2) $Ventas(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  R3) $Ruptura(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  R4) $Produ(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  R5) $Uso(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  R6) $0 \leq MachProd(< i,j,k >) \leq 1 \quad \forall < i,j,k > \in PosProd$

  R7) $CAlmac \geq 0$

  R8) $CProdu \geq 0$

  R9) $CRuptura \geq 0$

  R10) $Ingreso \geq 0$

  La lógica detrás de estas restricciones es un tanto trivial, ya que es evidente que no se pueden tener stocks o ventas negativas. La única que merece una mención aparte es la restricción de MachProd que, al ser una proporción, debe ser un número en el intervalo [0,1].

- *Continuidad del Stock:* En todas las funciones de decisión que dependan del tiempo, se debe introducir una ecuación de continuidad que relacione la 'utem i' con alguna de las demás, generalmente la 'utem i-1'. Más aún, debido a la dependencia temporal de estas funciones, generalmente se debe realizar una restricción aparte para la primera componente (ya que no existe la componente '0') y una general para todo el resto. En este caso, el

stock al final de la 'utem j' puede ser pensado como el stock al finalizar la 'utem j-1' más lo producido durante la 'utem j' menos lo vendido y menos lo utilizado como producto elaborado durante dicha 'utem'. Escribiendo esto formalmente:

R11) $Stock(i,1) = Produ(i,1) - Ventas(i,1) - Uso(i,1) \quad \forall\, i \in Tot_{Productos}$

R12) $Stock(i,j) = Stock(i,j-1) + Produ(i,j) - Ventas(i,j) - Uso(i,j)$
$\forall\, i \in Tot_{Productos}, j \in RHorizonte : j > 1$

- *Continuidad de la demanda:* Esta restricción debe reflejar la naturaleza acumulativa de la demanda no satisfecha. Escrito formalmente:

R13) $ruptura(i,1) + ventas(i,1) = DEMANDA(i) \quad \forall\, i \in Tot_{Productos}$

R14) $ruptura(i,j) + ventas(i,j) = DEMANDA(i) + ruptura(i,j-1)$
$\forall\, i \in Tot_{Productos}, j \in RHorizonte : j > 1$

- *Usos como Semi Elaborado:* Con la producción de todos los productos excepto aquellos en la primera etapa productiva, se consumen otros materiales como semi elaborados. Esta relación debe formalizarse en una restricción como la siguiente:

R15) $Uso(i,j) = \sum_{h \in T(i)} Produ(h,j) * BOM(i,h)$

Donde T(i) es el conjunto de productos que requieren del semi elaborado 'i' para su producción.

- *Continuidad de la Producción:* Se debe crear una restricción que vincule el tiempo de trabajo de las máquinas, es decir, MachProd con la producción obtenida, o sea, Produ. Dicha restricción se explicita a continuación:

R16)

$$Produ(i,k) = \sum_{j \in Tot_{Maquinas}} MachProd(< i,j,k >) * MATERIALPROD_1(j,i)$$
$$\forall\, i \in Tot_{Productos}, k \in RHorizonte$$

- *Límite Temporal de las Máquinas:* Para una máquina y una 'utem' dada, la suma de MachProd sobre todos los productos debe ser menor o igual a 1. De lo contrario, se estaría en la situación en que durante una 'utem', una máquina trabajó más de una 'utem' lo que viola leyes físicas básicas. Sería algo análogo a que en el transcurso de una hora, una máquina trabaje 70 minutos, lo que es evidentemente imposible. Por lo tanto:

  R17)

$$\sum_{j \epsilon Tot_{Productos}} MachProd(< i,j,k >) \leq 1 \;\; \forall\, i \epsilon Tot_{Maquinas}, k \epsilon\, RHorizonte$$

  Notar que la discretización del tiempo no impide que, durante una 'utem' dada, una máquina trabaje en más de un producto.

- *Capacidad Temporal de las Máquinas:* Ya sea por mantenimiento preventivo, por subalquiler de los bienes de uso u otras razones, es de esperar que las 'utem de máquina' disponibles para producir sean inferiores a las del horizonte de estudio. Por lo tanto, más allá de la restricción de límite temporal impuesto por R17, se deberá definir una restricción de capacidad temporal como la siguiente:

  R18)

$$\sum_{j \epsilon Tot_{Productos}} \sum_{k \epsilon RHorizonte} MachProd(< i,j,k >) \leq CAPACIDAD(i)$$
$$\forall i \epsilon Tot_{Maquinas}$$

- *Costo de Almacenamiento:* El costo de almacenamiento se calcula como en la ecuación (2) pero al haber discretizado el tiempo se aproxima la integral con la regla de los trapecios.

  R19)

$$\sum_{i \epsilon Tot_{Productos}} \left(\sum_{k=1}^{Horizonte-1} Stock(i,k) * MATERIALDEF_2(i) \right) + Stock(i, Horizonte) * 0{,}5 * MATERIALDEF_2(i)$$

  Notar que el stock al comienzo de la primera 'utem' es nulo, por eso no aparece en la restricción.

- *Costo de Ruptura:* El costo de ruptura se calcula como en la ecuación (3) pero haciendo la misma aproximación para la integral que en el caso anterior.

  R20)

$$\sum_{i \in Tot_{Productos}} \left(\sum_{k=1}^{Horizonte-1} Ruptura(i,k) * MATERIALDEF_3(i) \right) + Ruptura(i, Horizonte) * 0{,}5 * MATERIALDEF_3(i)$$

- *Costo de Producción:* Para calcular el costo de producción se aplica la ecuación (1) de la definición del problema.

  R21)

$$\sum_{<i,j,k> \in PosProd} MachProd(< i,j,k >) * MATERIALPROD_1(i,j) * MATERIALPROD_2(i,j)$$

- *Ingreso:* El ingreso se calcula a partir de la ecuación (4) de la definición del problema.

  R22)

$$\sum_{i \in Tot_{Productos}} \sum_{k=1}^{Horizonte} Ventas(i,k) * MATERIALDEF_1(i)$$

En este punto, con las 22 restricciones que determinan el espacio factible para la solución del problema, resulta interesante volver sobre las cuatro restricciones planteadas en la definición del problema y observar si se están cumpliendo. En primer lugar, la restricción de '*Capacidad de Máquinas'* está explícitamente representada por R18. En segundo lugar, el *'Diseño de Máquinas'* queda cubierto con la creación del conjunto PosProd, es decir, es imposible que una máquina produzca un material para el cual no está diseñada ya que la función de decisión MachProd está definida únicamente para aquellas combinaciones de máquina y producto válidas. En tercer lugar, el *'límite de ventas'* se asegura por la combinación de R3, R13 y R14. Es decir, si las ventas superasen su límite natural, entonces se tendría ruptura negativa, lo que viola R3. Por lo tanto, esto es imposible. Por último, con la *'necesidad de semi elaborados'* ocurre algo similar al combinar las restricciones R1, R11, R12 y R15. Si hay un exceso en la producción de un material, se tendrá stock negativo de sus componentes lo que viola la restricción R1. Por lo tanto, el exceso inicial es imposible.

Con esto se completa la resolución teórica del PAPP, permitiendo su ejecución computacional y el análisis de resultados.

3.5 Aplicación de la solución del PAPP

Para aplicar la solución desarrollada, se requiere un software de optimización matemática. En el presente trabajo se utilizará el *IBM ILOG CPLEX* pero cualquier aplicación podría ser útil. Las diferencias entre aplicaciones se deben principalmente al lenguaje que cada uno utiliza y al poder del motor de resolución.

En el apéndice A (9.2) se puede encontrar un código de aplicación de la solución del PAPP en *IBM ILOG CPLEX (CPLEX* de ahora en más). El lenguaje es intuitivo en su mayor parte por lo que permitirá su entendimiento aún sin la comprensión de todos los detalles. Más aún, para facilitar la tarea, se marcaron las 22 restricciones sobre el código (con marcas que van desde R1 hasta R22).

Con el objetivo de realizar un análisis, se aplicó la solución del apéndice A (9.2) a un conjunto de datos de entrada generado *ad-hoc.* Dicho conjunto de datos se ajustó al formato necesario para su compatibilidad con el apéndice A (9.2) y puede ser consultado en el apéndice B (10.1).

A grandes rasgos, el problema consiste en un horizonte de programación de 200 ‘utem’, dos etapas productivas con un total de 3 máquinas. Dos productos semi elaborados y seis productos finales.

Si bien no es el objetivo del presente trabajo, se mencionarán posibles orígenes de la información de entrada a ingresar en el caso de una fábrica real. El número de etapas productivas está claramente determinado por el proceso productivo elegido y la cantidad de máquinas en cada etapa depende de la tecnología empleada y de decisiones gerenciales. De todas formas, esta información es de fácil relevamiento. La lista de materiales (*Bill of Materials*) depende del proceso empleado. Sin embargo, en muchos casos, resulta una buena idea basarse en históricos de consumo en lugar de en los estándares del proceso. Esto es especialmente cierto cuando se utilizan medidas continuas de los materiales (litros, kilos, etc.). Con el ritmo de producción de cada máquina ocurre algo similar: más allá de la especificación del fabricante, conviene basarse en históricos que consideran estadísticamente ineficiencias humanas, tecnológicas y ambientales. Con los costos de producción, más allá de los históricos, uno requiere criterios de prorrateo de los costos generales de fabricación y, depende del nivel de interés del usuario la profundidad con la que se puede tratar este tema. Las capacidades de las máquinas se pueden calcular restando del horizonte el tiempo necesario para el mantenimiento preventivo, o, en caso en que no exista dicho mantenimiento, se

pueden utilizar históricos para conseguir el porcentaje de posible actividad de cada máquina. Para los precios de los materiales basta con observar el mercado y tomar algún criterio de unificación en caso de que los precios varíen con el cliente (por ejemplo, promedio ponderado por el volumen de compra). El costo de almacenamiento y de ruptura es más complicado de determinar, sobretodo el segundo ya que se deben evaluar conceptos intangibles como la pérdida de imagen. Los factores que determinan estos dos costos ya fueron discutidos en la descripción del PAPP y existen fórmulas para estimar dichos costos a partir de parámetros de cuantificación más sencilla. De todos modos, el desarrollo de estos conceptos excede el alcance del presente trabajo en esa área. La predicción de la demanda se puede realizar con un análisis de tendencia de históricos, teniendo en cuenta la combinación de una tendencia macro con estacionalidad y efectos aleatorios. Por último, el horizonte de estudio depende exclusivamente del objetivo del usuario a la hora de utilizar esta herramienta.

Una vez cargada la información de entrada, se utiliza algún método (en la próxima sección se discuten los métodos posibles) para hallar la solución óptima y extraer información útil.

Los resultados de este ejemplo se pueden observar en la tabla 3.4.

Beneficio	Ingreso	C. de Almac.	C. de Produ.	C. de Ruptura
142.240,7	188.171,1	0	41.979,7	3.950,7

Tabla 3.4: Beneficio, ingreso y costos para el conjunto de datos ingresado

En este punto, es esencial comprender el alcance del modelo descripto. ¿Es realmente representativo el valor del funcional maximizado? La respuesta es un rotundo no. Esto se debe a las múltiples limitaciones del modelo matemático que hacen que difiera significativamente de la realidad. El primer indicio a detectar es el costo de almacenamiento nulo, una situación poco realista. Esto se debe a la combinación de dos efectos: en primer lugar, no existe tiempo de *setup* cuando una máquina pasa de producir un material a otro; en segundo lugar, no existe tiempo de transporte (*leadtime*) entre las etapas productivas. Como consecuencia, cada máquina produce tres o cuatro productos en una misma 'utem', una situación poco real que lleva justamente al costo de almacenamiento nulo.

Más aún, este modelo presenta otras limitaciones como una demanda determinada y fija a lo largo del tiempo, tiempos y costos de producción independientes del tamaño del lote, entre otros. Muchas de estas deficiencias se irán superando a lo largo del presente trabajo, acercándose sistemáticamente a la realidad.

Ante esta situación, suena lógico cuestionar la utilidad de la solución del PAPP. La respuesta es la siguiente: **la solución al PAPP provee información exclusivamente cualitativa**.

A continuación se presenta información útil que puede ser extraída de la solución del PAPP. El siguiente listado no intenta ser una lista acabada, por el contrario se busca demostrar el amplio espectro de la información que se puede extraer y su utilidad, mediante cinco ejemplos.

a) *Unidades Totales de Producto Terminado*

A continuación (Figura 3.3) se grafican el total de las unidades producidas de cada producto terminado:

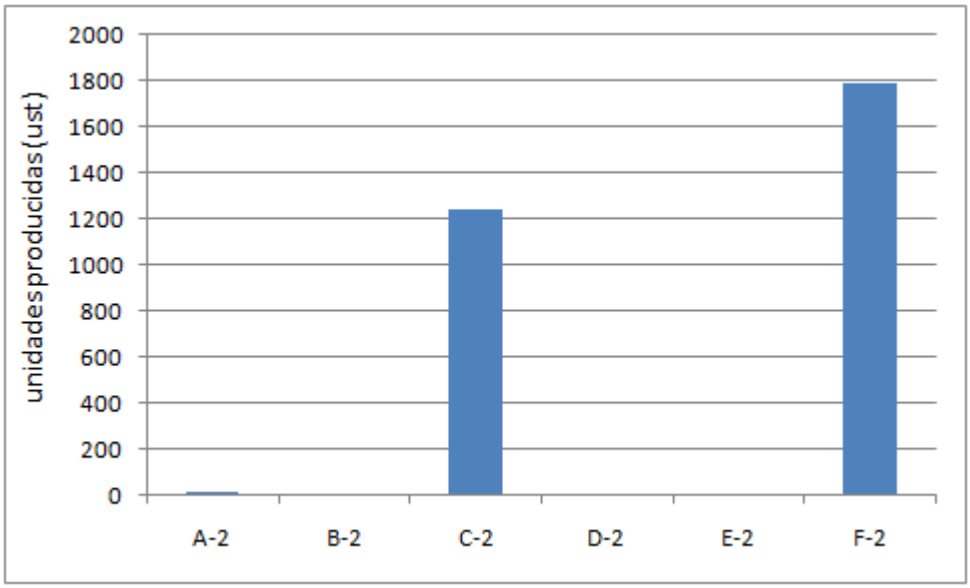

Figura 3.3: Unidades totales producidas en función del producto terminado

Si bien los valores absolutos poca información pueden aportar debido a las limitaciones antes mencionadas, la información relativa es de elevada riqueza para decisiones gerenciales. En este ejemplo (un tanto extremo, por cierto) es fácil determinar que los productos importantes son el “C-2” y el “F-2”, por lo tanto son aquellos donde se debe enfocar la energía. Por ejemplo, si un proveedor de un insumo para producir “A-2” amenaza con cortar el suministro, no se debería estar dispuesto a ofrecer lo mismo que si ocurre con un proveedor de “F-2”. Es decir, estos datos ayudan a asignar importancia relativa a los productos para tomar decisiones gerenciales eficientes.

b) *Costo de introducción de un producto*

En este punto, se puede recurrir al análisis de sensibilidad de la solución hallada. Para ilustrar esta idea, se supondrá que el departamento de marketing exige que se produzca ya sea el producto “B-2” o el “D-2” ambos fuera de la solución óptima. Más allá del valor absoluto, basta con notar que el costo reducido de introducir “B-2” es menos de la mitad del de “D-2” (67 contra 159) para suponer que la introducción de “B-2” desvía en menor medida la nueva solución de la óptima, productivamente hablando. A partir de allí, se pueden realizar análisis de mayor profundidad.

c) *Utilización de Máquinas*

A partir del estudio de la figura 3.4 sobre las ‘utem’ utilizadas de cada máquina, se pueden extraer algunas conclusiones.

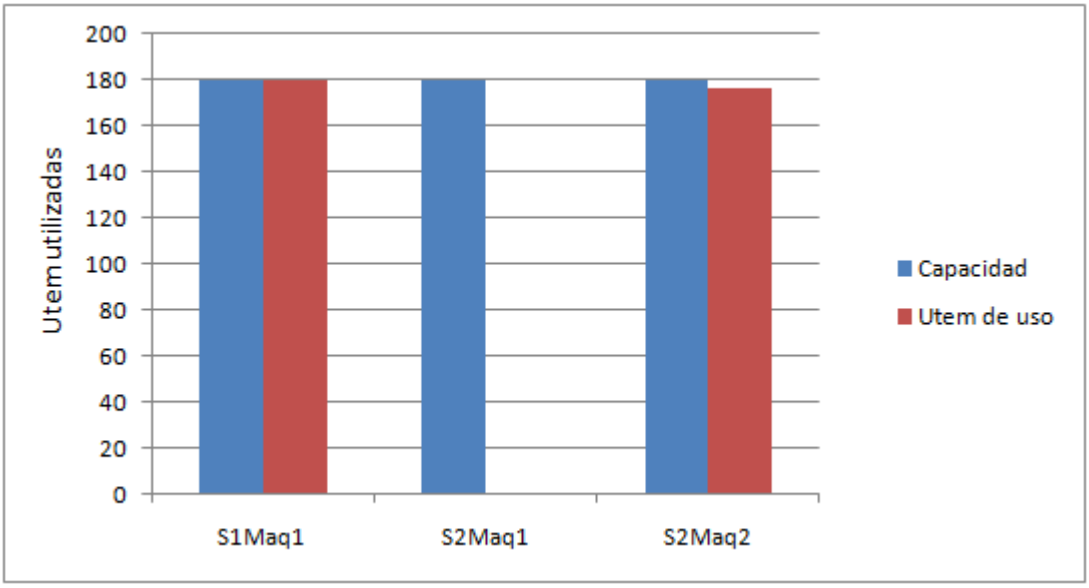

Figura 3.4: Capacidad y utilización de cada máquina

Al observar este gráfico, la situación se presenta con una claridad notable. La única máquina de la primera etapa productiva (“S1Maq1”) se encuentra trabajando al máximo de su capacidad. Por otra parte, una de las máquinas de la segunda etapa consume todos los semi elaborados producidos en la primera etapa productiva, dejando absolutamente sin utilidad a la otra máquina de la segunda etapa (“S2Maq1”). A partir de este análisis, queda claro que es necesario aumentar la capacidad productiva de la primera etapa para poder dar uso a la máquina que se encuentra totalmente inutilizada. Nuevamente, hay que comprender que en la situación real puede ser que la utilización de “S2Maq1” no

sea nula como en este modelo, pero sin duda se presenta el fenómeno de subutilización debido a una capacidad deficiente en la etapa precedente.

d) *Valor de la 'utem' de máquina adicional*

Nuevamente, se puede recurrir al análisis de sensibilidad pero en este caso sobre la restricción de capacidad de máquina. En este ejemplo, no agrega demasiada información ya que tan sólo una máquina no presente holgura. Sin embargo, a partir de la comparación de los valores duales de aquellas máquinas sin holgura, se puede determinar el valor relativo de un aumento de capacidad en una u otra máquina. Este aumento de capacidad puede pensarse como una adquisición de máquinas o como una tercerización de parte de la producción. Por lo tanto, el valor dual de la restricción de capacidad puede utilizarse como información gerencial a la hora de subcontratar otra fábrica o comprar una máquina.

e) *Elección de la máquina en cual producir*

De la descripción del PAPP se desprende que un producto dado puede ser producido por más de una máquina con ritmo y costo diferentes. En ocasiones es sencillo predecir en qué máquina conviene realizar esto, sobretodo en los casos en que aquella de menor costo tiene mayor ritmo. Sin embargo, esto no siempre es así. Por lo tanto, más allá de los valores absolutos, analizar qué proporción de cada producto se fabrica en cada máquina arroja información de utilidad valiosa a la hora de tomar decisiones de producción.

3.6 Algoritmos de resolución para problemas continuos

En el presente apartado se compararán tres métodos de optimización matemática para problemas de programación lineal continua: Simplex Primal, Simplex Dual y Barrera.

Para dejar en claro en qué momento se aplica cualquiera de estos tres algoritmos, se realizó el esquema de la figura 3.5:

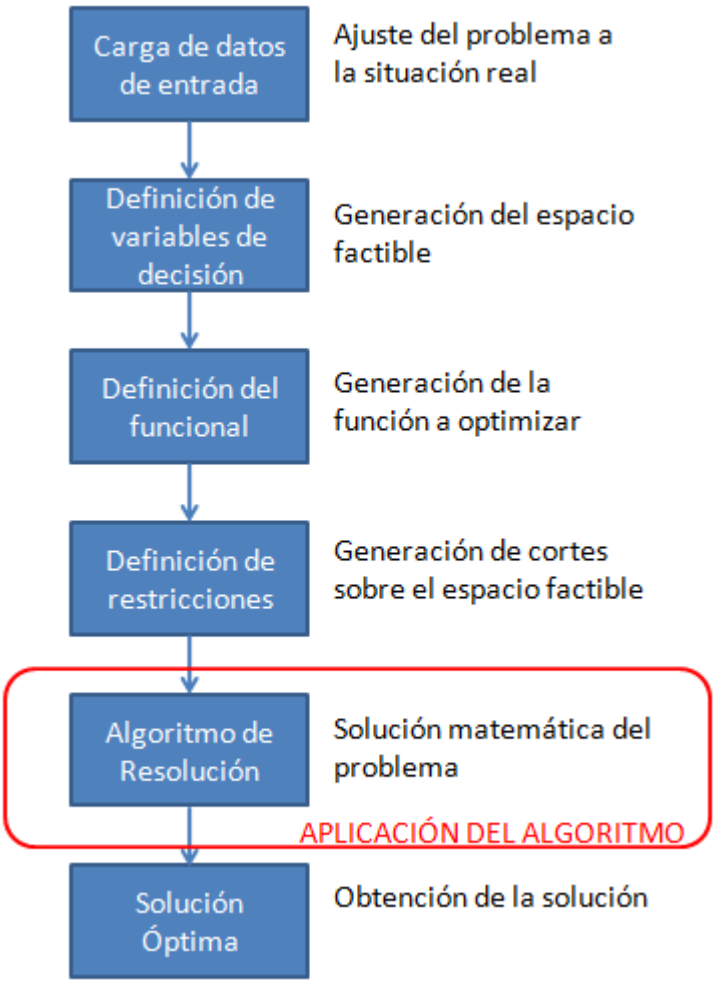

Figura 3.5: Esquema de resolución general de un problema de optimización

De más está decir que los tres algoritmos devuelven la misma solución para un mismo problema ya que ésta no depende del método utilizado. Sin embargo, lo que varía es el tiempo de resolución de cada método dependiendo de la naturaleza del problema.

Por consiguiente, se buscará el método que mejor se ajuste a la resolución de un problema de programación de producción general, a partir de características distintivas de éste.

Si bien no se explicarán los tres métodos con la profundidad suficiente para comprenderlos desde la rigurosidad matemática, se presentarán algunas características diferenciales de cada uno.

El Simplex Primal es un algoritmo de dos fases. En la primera fase, busca una solución factible, pero no óptima del problema. Luego, iterativamente mejora su optimalidad hasta arribar a la solución óptima del problema. Gráficamente,

comienza en un vértice del polígono de soluciones factibles y luego se mueve hacia vértices adyacentes optimizando el funcional.

El Simplex Dual también es un algoritmo de dos fases. Sin embargo, en este caso se selecciona un punto de partida óptimo pero no factible. Luego, iterativamente se mejora la factibilidad hasta llegar a un punto que es tanto óptimo como factible, es decir, la solución óptima. Gráficamente, comienza fuera del polígono de soluciones factibles y se acerca iterativamente al vértice de la solución óptima.

El algoritmo de Barrera no presenta las dos etapas diferenciadas (elección del punto de partida y acercamiento iterativo a la solución). Por el contrario, selecciona múltiples puntos, tanto dentro como fuera del espacio de soluciones factibles generando una serie de puntos convergentes a la solución óptima.

La figura 3.6 ilustra los tres algoritmos de manera gráfica en un espacio de decisión bidimensional con algunas restricciones lineales.

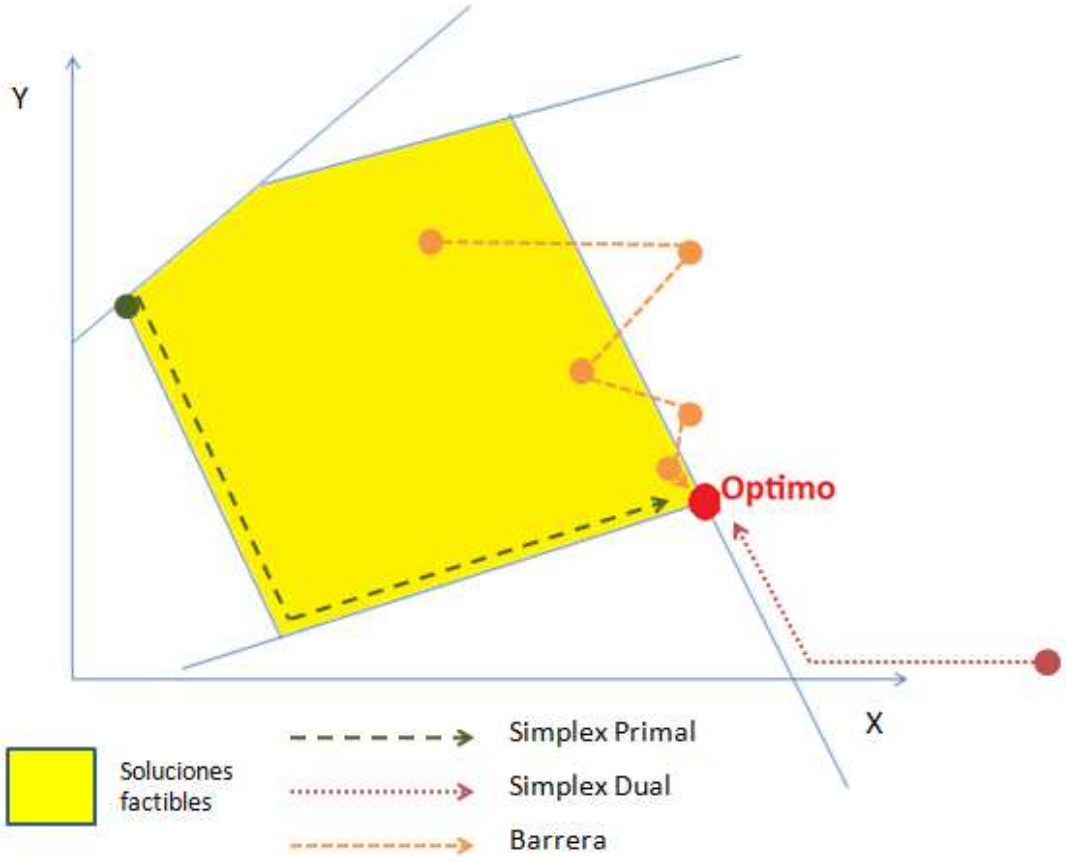

Figura 3.6: Descripción gráfica del Simplex Primal, el Simplex Dual y Barrera

A continuación, se desarrollan algunos conceptos que pueden ser tomados como generalidades pero siempre teniendo en cuenta que pueden no cumplirse para

algunos conjuntos de datos y estructuras determinadas. Sin embargo, para la mayoría de los casos son reglas que vale la pena seguir como se mostrará con el ejemplo.

En primer lugar, entre los métodos Simplex, conviene utilizar el Simplex Dual cuando el número de restricciones supera al número de variables de decisión. Esto se debe a que en estos casos es más sencillo encontrar el punto de partida fuera del espacio de soluciones factibles y el camino hacia la solución óptima requiere de menos iteraciones. En el caso del PAPP y las subsiguientes modificaciones del problema de programación de producción, el número de restricciones supera al de las variables de decisión, siendo el Simplex Dual el preferido.

Por otro lado, el algoritmo de Barrera, es generalmente más eficiente para problemas de gran tamaño (miles de variables de decisión y restricciones) y que presenten cierto grado de esparcimiento. En los problemas aquí estudiados, se superan fácilmente las mil variables de decisión y restricciones. Notar que aunque se enumeraron 22 restricciones, muchas de ellas se encuentran definidas para todo un conjunto, lo que aumenta radicalmente el número de restricciones “reales” a considerar.

En rigor, no se puede asegurar que el algoritmo de Barrera será siempre más rápido que los otros dos para problemas de gran envergadura y con alta proporción de ceros en sus columnas ya que si se estudia con un poco más de detenimiento su comportamiento (algo que no se extenderá en el presente trabajo), se comprende que en cada iteración calcula el factor de Cholesky de la matriz A^TA donde A es la matriz de restricciones que cumple Ax=b. Por lo tanto, puede ocurrir que el factor de Cholesky requiera de mucho cálculo y los otros algoritmos lo sobrepasen en performance. Sin embargo, es esperable que en la generalidad de los casos, el algoritmo de Barrera sea la mejor opción para resolver un PAPP.

En la figura 3.7, se analiza el tiempo de resolución del PAPP ejemplo, apéndice B (10.1), con los tres métodos. Además, se varía la dimensión del problema modificando el horizonte de estudio para analizar la dependencia del tiempo de resolución con la dimensión para los tres métodos.

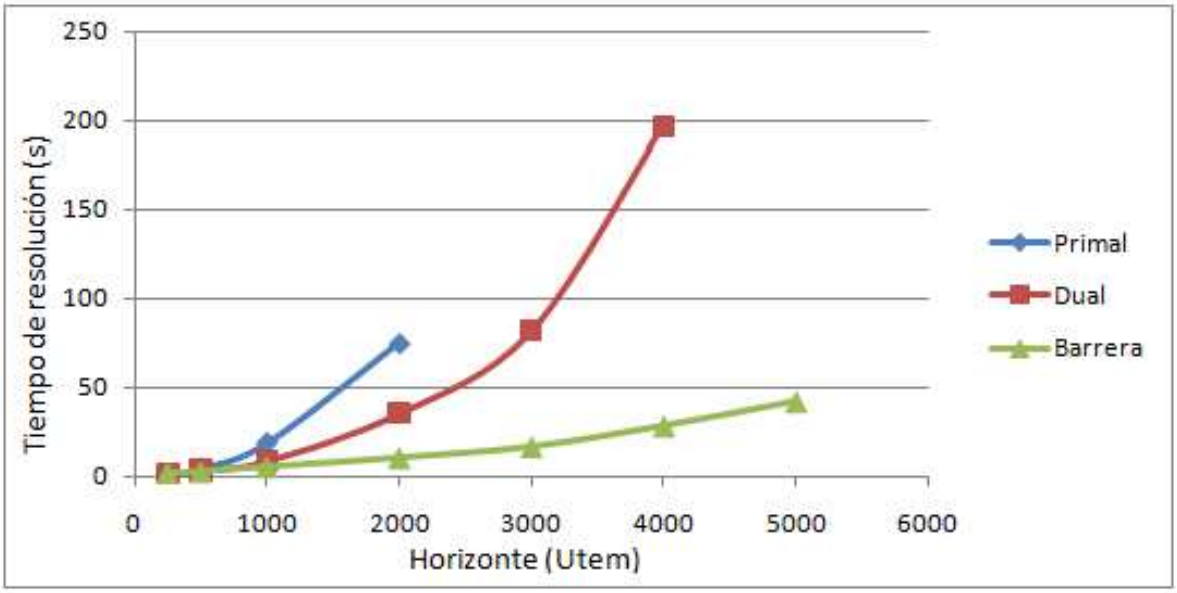

Figura 3.7: Dependencia del tiempo de resolución del PAPP con la dimensión del problema para los tres métodos

Como era de esperarse, para dimensiones pequeñas, la elección del método es indiferente ya que las diferencias son del orden de décimas de segundo. Sin embargo, al aumentar la dimensión del problema, se observa que los tiempos de resolución aumentan de diferente forma para los tres métodos respetando las reglas expuestas anteriormente. Aquí se puede observar claramente que el algoritmo de optimización por Barrera es más eficiente que los otros dos.

3.7 Naturaleza de las unidades físicas

En este punto, se podría argumentar justificadamente que si las 'ust' hasta aquí mencionadas no son de carácter continuo (kilos, litros) sino discreto (por ejemplo, cantidad de baterías producidas) entonces toda la definición continua del PAPP y los algoritmos estudiados en la sección precedente carecen de sentido ya que, inevitablemente, la variable 'unidades producidas' debe ser entera y no se podría aplicar la resolución con programación lineal continua. Esta interesante disquisición será tratada en esta sección.

Para ilustrar el concepto, se considerará un problema más sencillo que el PAPP:

"Se posee una planta que puede fabricar 'd' productos finales utilizando una combinación de 'd' materias primas (sólo una etapa productiva). Cada producto final 'i' (i = 1..d) aporta un margen 'm_i' al beneficio total de la empresa. El objetivo

es, dado un stock de materia prima, encontrar la cantidad a fabricar de cada producto final para maximizar el beneficio"

A continuación se formula matemáticamente:
Siendo:

- $Compo_{ij}$ (i<=d, j<=d) la cantidad de materia prima 'j' necesaria para una unidad del producto terminado 'i'
- $Margen_i$ (i<=d) el margen aportado por el producto 'i' al beneficio de la compañía
- $Cantidad_j$ (j<=d) el stock utilizable de la materia prima 'j'
- $Producir_i$ (i<=d) la producción a realizar del producto final 'i' (Variable de decisión)

Maximizar:

$$\sum_{i=1}^{d} Producir_i * Margen_i$$

Sujeto a:

$$\sum_{i=1}^{d} Producir_i * Compo_{ij} \leq Cantidad_j \qquad \forall j = 1 \dots d$$

Notar que este es un problema muy sencillo de comprender cuya dimensión puede variar fácilmente modificando el parámetro de dimensionamiento 'd'.

La cuestión es la naturaleza de la variable de decisión "$Producir_i$", ¿debe ser entera o continua? Una primera aproximación es la siguiente: si la unidad física que está siendo representada es continua entonces la variable debe serlo, de lo contrario, no. Con este criterio, si los productos terminados son toneladas de granos, la variable de decisión debe ser continua. Si, por el contrario, los productos terminados son martillos la variable debe ser discreta (cantidad de martillos) ya que es imposible fabricar una cantidad decimal de martillos.

Si ahora se analizan los algoritmos matemáticos que se utilizan para la resolución de ambos casos, se cuenta con información adicional para decidir. En el caso en que las variables de decisión son continuas, se utilizan alguno de los algoritmos mencionados en la sección precedente. Sin embargo, con las variables discretas se utilizan algoritmos de corte (como el Branch & Bound), que serán explicados

más adelante, pero que tienen un tiempo de resolución notablemente mayor que los primeros.

En este punto, se podría tomar la siguiente posición: resolver el problema con variables continuas y luego tomar los enteros más próximos como solución del problema entero. Esto no es necesariamente la mejor decisión. En primer lugar, la solución con los enteros más próximos no es necesariamente factible. En segundo lugar, en caso de ser factible, no es necesariamente óptima. Notar que si en lugar de elegir los enteros más próximos se elige la función 'piso' de la solución continua, se asegura factibilidad para este ejemplo en particular pero se sigue sin asegurar optimalidad.

En la figura 3.8 se esquematiza dicho concepto para un espacio factible bidimensional:

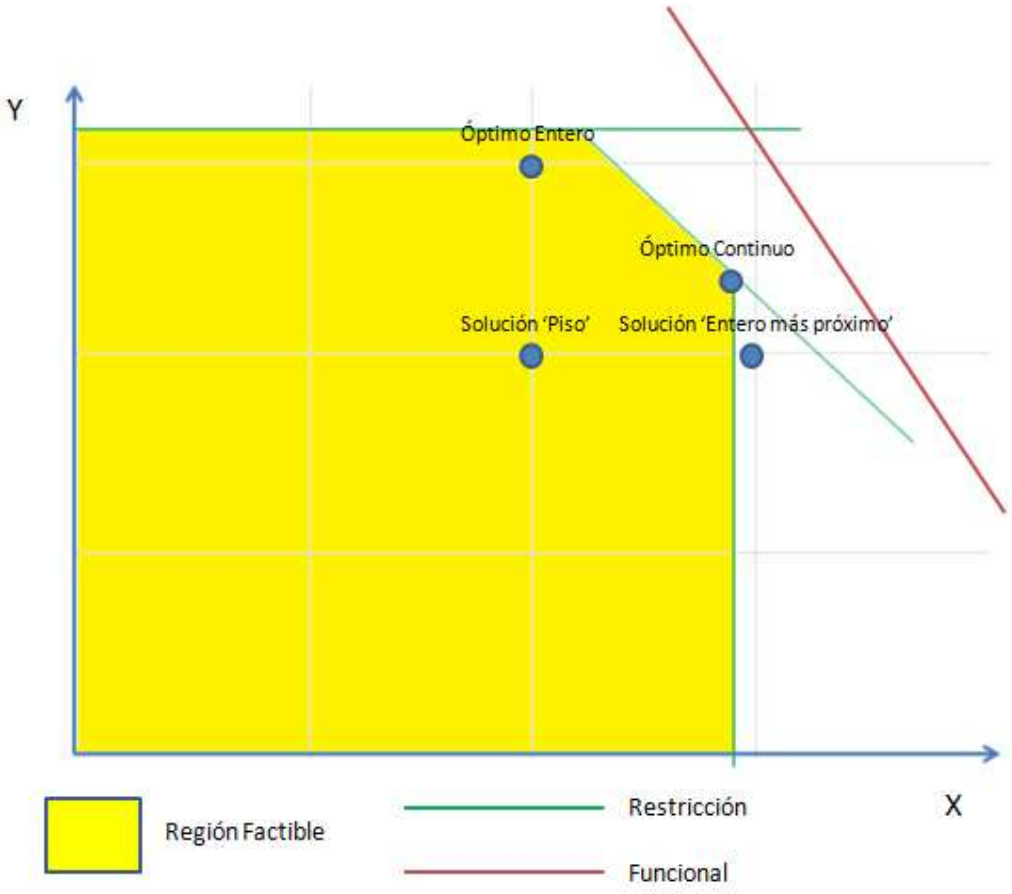

Figura 3.8: Soluciones 'Continua' y 'Entera' para un problema lineal de dos variables

Si se intenta resolver el problema representado por la figura 3.8 con un algoritmo continuo, entonces se arribaría al punto 'Óptimo Continuo' en un tiempo

relativamente pequeño. Si luego se intenta introducir al entero más próximo como solución, éste que cae fuera de la región factible, por lo que necesariamente viola alguna restricción. Si, por el contrario, se toma el piso de la variable X y la variable Y de la solución continua como solución entera, entonces se arribaría al punto "Solución 'Piso'" que es una solución válida. Sin embargo, no es la solución óptima en el espacio de las variables enteras como se puede observar en el esquema. Pero, se debe tener en cuenta que para arribar a la solución "Óptima Entera" se debe utilizar un algoritmo del tipo Branch & Bound cuyo tiempo de resolución es mayor.

Así se arriba a la esencia de la cuestión. El criterio de decisión debe ser el siguiente: si la diferencia temporal entre las resoluciones de ambos métodos es más valiosa que la diferencia obtenida en el funcional, entonces utilizar algoritmos continuos. De lo contrario, utilizar algoritmos discretos.

Llevando esto a la práctica. Si se considera una fábrica de golosinas, que produce distintos tipos de caramelos, chupetines y chocolates, se puede observar que las unidades son físicamente discretas. Sin embargo, a la hora de planificar su producción, no tiene sentido utilizar variables de decisión discretas ya que por la mejora de algunos pesos en el beneficio, la solución puede demorar varias horas. Por el contrario, si se considera una fábrica de aviones comerciales, sin duda se deben utilizar variables discretas que maximicen el beneficio para los recursos dados. En este caso, la producción de un avión más o menos impacta radicalmente en el beneficio, mucho más que la pérdida de tiempo durante la planificación.

Para concluir el tema, se realizaron dos demostraciones. En primer lugar se resolvió el problema planteado al comienzo de la sección para d=10 con algoritmos continuos y discretos. Para consultar su aplicación en *CPLEX,* referirse al apéndice A (9.3.1). Los beneficios para los distintos puntos esquematizados en la figura 3.8 se enuncian en la tabla 3.5 a continuación:

Óptimo Continuo	Solución Entera más próxima	Solución 'Piso'	Óptimo Entero
54.310,6	No Factible	54.164	54.294

Tabla 3.5: Comparación de funcionales para distintas soluciones

En este caso se observa una diferencia del 0,2 % entre la solución 'Piso' de la continua y la solución entera. Nuevamente, dependiendo de la naturaleza de lo que se fabrique y del tiempo disponible se elegirá una solución o la otra.

En segundo lugar, se resolvió el problema planteado para distintos parámetros de dimensionamiento 'd' con el algoritmo de resolución continua y discreta. Cabe

aclarar aquí (aunque se explicará con detalle más adelante en el trabajo) que para el algoritmo discreto se consideró “tiempo de resolución” al necesario para alcanzar la solución óptima y probar su optimalidad.

La aplicación en *CPLEX* se puede encontrar en el apéndice A (9.3.2) y los resultados se observan en la figura 3.9:

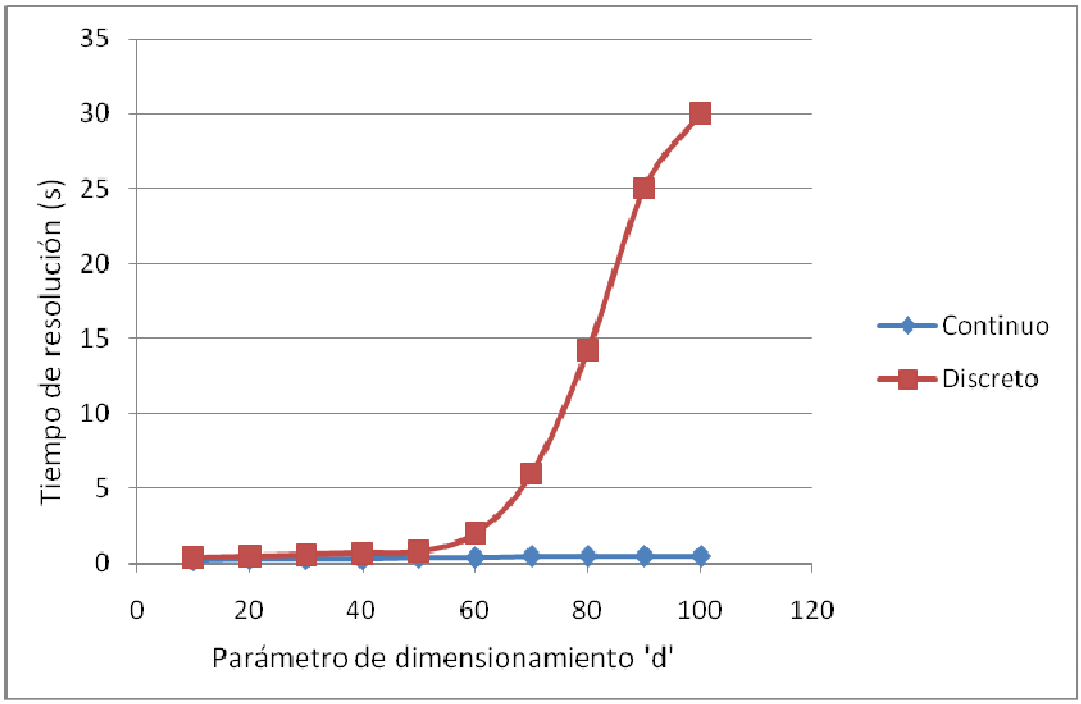

Figura 3.9: Tiempos de resolución para distintas dimensiones del problema para ambos métodos

Se puede observar que para el método discreto el tiempo parece crecer de manera exponencial (más adelante esto se justificará) y, además, tiene una fuerte dependencia sobre los datos de entrada (mas allá de la dimensión). Esto último no se puede observar del gráfico, pero se probará más adelante. De todas formas, es interesante notar que para una dimensión de d=100, el algoritmo discreto demora unas 30 veces más que el continuo. En este marco, cobra sentido comenzar con la resolución en el espacio continuo para luego buscar la Solución ‘Piso’.

3.8 Múltiples épocas dentro de un PAPP

Previamente se mencionó, como una de las limitaciones del PAPP, que suponía una demanda determinada y constante de cada producto en el mercado. En este apartado se discute una variación de esta característica que se denominará “Multi-época”. Es decir, la duración total “Horizonte” se dividirá en ‘n’ sub períodos

denominados épocas que se diferenciarán del resto en dos aspectos fundamentales. Por un lado, la demanda de cada producto es constante dentro de una época pero puede variar entre épocas. Por otro lado, la capacidad horaria de cada máquina está dada en función de la época, en lugar de ser una capacidad total para el horizonte de estudio.

La división en épocas es una característica que acerca el modelo a la realidad. Por ejemplo, si el "horizonte" es un año, tiene sentido dividir al año en épocas (las 4 estaciones o los 12 meses) ya que la demanda puede sufrir fuerte influencia de estacionalidades. Como otro ejemplo, si el "horizonte" es una semana, también resulta lógica la división en épocas (1 época para cada día o una época para los días de semana y otra para los fines de semana).

Como es natural, el planteo del problema se diferencia ligeramente del descripto hasta aquí. En particular, las funciones de entrada DEMANDA y CAPACIDAD dejan de ser funciones de una variable para pasar a depender también de la época. Más aún, las respectivas restricciones R13, R14 y R18 se deben replantear teniendo en cuenta esta nueva dimensión de las funciones mencionadas. Sin embargo, en esta sección no se profundizará en los aspectos formales del planteo. Por el contrario, se intentará responder la siguiente pregunta, cuya respuesta será de utilidad más adelante:

"En un problema multi-época generalizado, ¿es razonable utilizar la optimización de cada época por separada como solución del problema global?"

La utilidad de la respuesta reside en que, para problemas más complejos que utilizan programación entera mixta, el tiempo de resolución crece exponencialmente como se pudo observar en la figura 3.9. Como consecuencia, resolver dos épocas de duración "d" por separado puede ser mucho más eficiente que resolver una época de duración "2d". Para responder dicha pregunta, se resolverá un problema cuyos datos de entrada se encuentran en el apéndice B (10.2) de manera global y separando por épocas y se compararán las soluciones.

El problema es similar al del apéndice B (10.1) pero el horizonte de 200 'utem' se dividió en cuatro épocas de 50 'utem' cada una. Notar que en los datos de demandas y capacidades de máquinas, se agregó una columna para indicar la época a la que se está refiriendo.

A continuación, en la tabla 3.6 se presentan los beneficios obtenidos con ambas estrategias:

Beneficio Optimización Global	Beneficio Optimización por Época
$378.093	$273.867

Tabla 3.6: Comparación de los beneficios con ambas estrategias de optimización

Más allá de los valores absolutos, que poseen poca información por separado, la relación entre ambos es de suma importancia. Se puede apreciar que la optimización separada por épocas, en este ejemplo, tiene un óptimo 28% inferior al óptimo global. Es evidente que esta cifra varía con la naturaleza de los datos.

Un breve análisis de los resultados en ambos casos, permite una mejor comprensión de la situación. En primer lugar, se analizarán en la figura 3.10 los beneficios de cada época utilizando ambas estrategias:

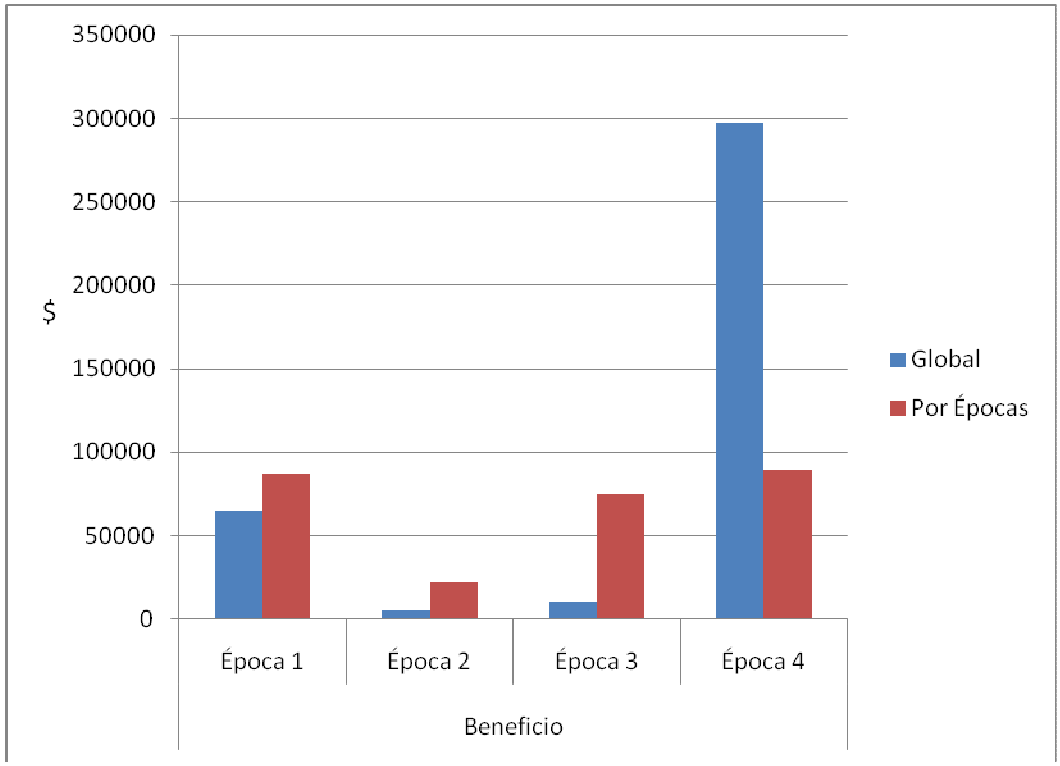

Figura 3.10: Beneficios por época para las dos estrategias de optimización

Un primer análisis arroja que la optimización por épocas obtiene mayor beneficios en las primeras tres épocas pero tiene una performance muy inferior en la última. De todas formas, la explicación no queda del todo clara a este nivel de agregación.

Por esta razón, se explicitará el beneficio en términos de los ingresos y costos (excluyendo al de ruptura por ser despreciable en este ejemplo) en la figura 3.11:

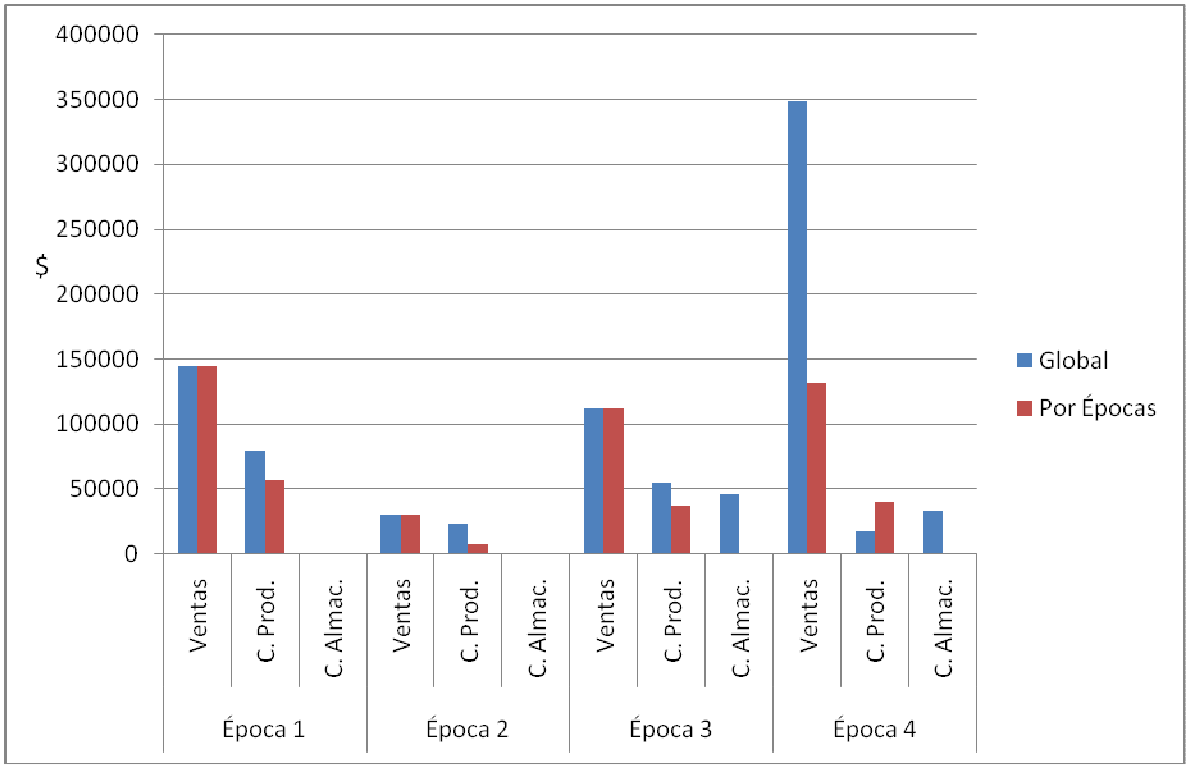

Figura 3.11: Ingresos y costos por época para las dos estrategias de optimización

En este caso, la explicación de las diferencias en los beneficios se hace evidente. En primer lugar, se debe observar que el plan de ventas es idéntico para ambas estrategias durante las primeras tres épocas. Por lo tanto, la diferencia en el beneficio se debe a un mayor costo (tanto de producción como de almacenamiento) por parte de la estrategia global. Este mayor costo durante las primeras tres épocas genera el stock necesario para poder alcanzar los elevados niveles de venta durante la última época.

Este ejemplo demuestra una diferencia evidente pero fundamental entre ambas estrategias. La estrategia global, desarrolla un programa de producción que, para cada época, analiza el impacto en el resto. Por esta razón, la estrategia global 'decide' producir de más y almacenar durante las primeras tres épocas porque 'sabe' que las ventas en la última época supondrán un beneficio mayor que el costo incurrido. La optimización por épocas, en cambio, sufre de *'miopía'* temporal. Con esto se intenta transmitir que esta segunda estrategia optimiza el beneficio de cada época por separado sin considerar, por ejemplo, almacenar para futuras ventas.

Se podría argumentar que, en ocasiones que se presenten estacionalidades menos marcadas, la performance de la estrategia por épocas sería comparable con la global. Esto es cierto. Sin embargo, las épocas surgen como necesidad de

plasmar la estacionalidad (ya sea en demanda o capacidad) en los datos de entrada. Por lo tanto, como situación general, se debería esperar cierto grado de estacionalidad entre épocas.

Finalmente, se puede contestar la pregunta planteada. No es razonable optimizar las épocas por separado ya que la utilidad de esta estrategia se reduce con el nivel de estacionalidad y, justamente, las épocas se introducen para representar ésta. Debe quedar claro que aquí se intenta resolver la cuestión para un problema general, lo que no quita que existan problemas con marcada estacionalidad cuya solución por épocas se aproxime a la solución global. Sin embargo, como el objetivo del presente trabajo es generar una herramienta lo más general posible, sería arriesgado adoptar una estrategia de optimización por épocas como válida.

4 PROBLEMA COMPLETO DE PROGRAMACIÓN DE LA PRODUCCIÓN

4.1 Introducción

En la presente sección, se completará el PAPP multi-época agregando diversas características al modelo matemático que lo acerquen a la realidad. Este nuevo problema a resolver se denominará "Problema Completo de Programación de la Producción" (PCPP). Es importante comprender que la denominación "*completo",* no presume indicar que el modelo es lo más cercano a la realidad posible. En cambio, con PCPP se indica que este es el modelo más completo que se tratará en el presente trabajo pero que presenta numerosas áreas de posible mejora. Algunas de éstas se discutirán más adelante en la sección de "Limitaciones y Futuros Pasos".

Las características novedosas del PCPP son las siguientes:

- *Lead-times* (tiempo de transporte de materiales) entre etapas productivas
- Tiempos y costos de *setup* al cambiar la producción de una máquina
- 'Inventarios objetivo' al finalizar el período de estudio

4.2 Descripción del problema

Está claro que una descripción completa del PCPP sería tanto tediosa como innecesaria ya que éste es una ampliación del PAPP. Por lo tanto, simplemente se explicarán aquellos aspectos diferenciales entre el PCPP y el PAPP, siendo éstos los que agregan valor práctico al primero. Sin embargo, a la hora de resumir estructuras de datos para facilitar su comprensión (por ejemplo, la información de entrada), se repetirá lo expuesto en el PAPP con el objetivo de contar con un fácil y único acceso a la información resumida del PCPP.

En primer lugar, se ampliarán las tres modificaciones mencionadas arriba, no desde su aplicación algorítmica (explicada más adelante) sino desde el punto de vista conceptual:

- *Lead-times* entre etapas productivas

  El PAPP suponía un abastecimiento instantáneo de las sucesivas etapas de producción. Es decir, siguiendo la nomenclatura del PAPP, si se considera el producto X ϵ $Pro_{i-(i+1)}$ entonces cada 'ust' de X producida en la etapa 'i' viaja con tiempo nulo a la etapa 'i+1'. Desde luego, esta suposición es poco realista.

Por lo tanto, el PCPP introduce el *lead-time* entre etapas. El modelo que se decidió seguir es un *lead-time* fijo a lo largo del horizonte de estudio que varía con la ruta a seguir. Es evidente que el *lead-time* entre dos etapas depende tanto del *Lay-Out* de la planta como de los medios de transporte de materiales de los que se dispone. De esta forma, el *lead-time* se puede ajustar a la realidad de cada planta, incrementando la flexibilidad del PCPP.

Cada producto semi elaborado se almacena en la etapa que va a ser utilizado. Es decir, continuando con el ejemplo del producto X, cada 'ust' del producto X se fabrica en la etapa 'i' y se transporta inmediatamente a la etapa 'i+1' en donde ingresa al almacén intermedio. Como consecuencia, se adoptará la siguiente convención: *los productos semi elaborados que se encuentren viajando entre etapas productivas no se consideran para el cálculo del costo de almacenamiento*.

El *lead-time* no se aplica al lote de producción en conjunto, sino que se aplica a cada 'ust' o fracción de ésta producida. Idealmente, el *lead-time* se aplicaría de manera continua a cada 'diferencial de cantidad' producido. De este modo, llamando 'L' al *lead-time* entre las etapas 'i' e 'i+1' del ejemplo anterior, se debería observar la correlación de la figura 4.1:

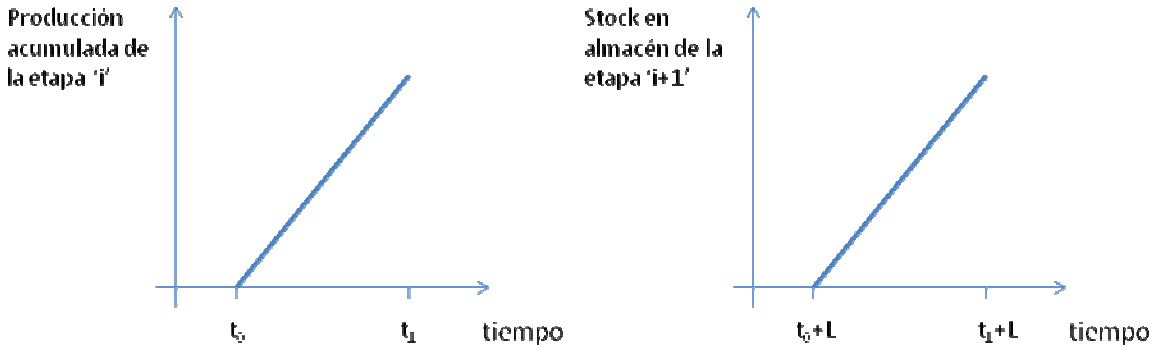

Figura 4.1: Característica ideal de continuidad del *lead-time* entre etapas

De todas formas, a la hora de representar el *lead-time* en la solución, se realizará una discretización temporal como la discutida en el PAPP.

- Tiempos y costos de *setup* al cambiar la producción de una máquina

Al analizar la información de salida del PAPP resuelto, una de las características que más contrasta con la realidad es el constante cambio en el material a producir por cada máquina. Esto se debe a que el cambio de materiales a producir en el PAPP no se penaliza desde ningún punto de vista. En el PCPP, en cambio, el cambio de materiales se penaliza con lo que

comúnmente se denomina *setup*. La penalización es doble, tanto temporal como monetaria. Es decir, la realización de un *setup* dado requiere de cierta cantidad de tiempo durante el cual la máquina está inutilizada (pero este tiempo consume capacidad horaria de la máquina) y además conlleva un cierto costo. Es evidente que estas suposiciones acercan el PCPP a la realidad. El tiempo de *setup* se puede asociar al lavado de las máquinas entre lotes, al cambio de matrices, al reemplazo de insumos consumibles, entre otros, dependiendo de la industria que se trate. El costo del *setup* se relaciona tanto con el sueldo de los operarios que lo realizan como con el costo de los materiales consumidos durante el proceso (por ejemplo, el material de limpieza o el agua).

En el modelo matemático del tiempo y costo de *setup* que se seguirá, éstos dependen de tres variables: la máquina, el producto que se deja de fabricar y el producto que se comienza a fabricar. Por consiguiente, la información de entrada que se requiere es numerosa y su fuente será, preferentemente, registros históricos de los procesos de *setup.* Como en el PAPP, el tiempo y costo de *setup* pueden ser nulos para una terna dada (máquina, producto, producto), pero no lo es en la generalidad de los casos. Una suposición que se realiza es que, si el tiempo de *setup* es nulo, entonces su costo también lo es. Por último, cabe destacar que el *setup* no es 'simétrico' en el sentido que una máquina dada, para pasar de fabricar el producto A al producto B no tarda ni cuesta lo mismo que para pasar del producto B al producto A.

De esta forma, en el PCPP se debe agregar un cuarto costo al funcional, el costo de *setup.* Conceptualmente, si se denomina mach_setup(i,x,y) a la cantidad de 'utem' que la máquina 'i' estuvo realizando *setup* para pasar de producir el producto 'x' al producto 'y' y $SETUP_2(i,x,y)$ al costo de cada 'utem' del correspondiente *setup,* entonces:

$$C.Setup = \sum_{i \in Tot_{Maquinas}} \sum_{x \in Tot_{Productos}} \sum_{\substack{y \in Tot_{Productos} \\ y \neq x}} mach\_setup(i,x,y) * SETUP_2(i.x.y) \qquad (6)$$

- 'Inventarios objetivo' al finalizar el período de estudio

Es lógico suponer que un programa de producción puede tener objetivos que vayan más allá del beneficio obtenido durante el horizonte de estudio. Por ejemplo, al optimizar la producción de un mes, se pueden imponer 'inventarios objetivo' al finalizar dicho período para cada uno de los productos. Dichos niveles de inventarios deberían ser los inventarios iniciales convenientes para

el siguiente período a optimizar. De todas formas, debe quedar claro que los 'inventarios objetivo' no se imponen como condiciones de borde a la solución. Por el contrario, se asigna un costo de incumplimiento del 'inventario objetivo' calculado como la diferencia entre el 'inventario objetivo' y el inventario final para cada producto multiplicado por un *factor de déficit*. Este es un costo más que se agrega al funcional y que se tiene en cuenta a la hora de optimizar el programa de producción. De esta forma, cuando el *factor de déficit* es nulo los 'inventarios objetivo' no se tienen en cuenta. Por el contrario, cuando este factor crece, van tomando cada vez más importancia hasta convertirse en restricciones a cumplir (siempre que sea factible) cuando el factor tiende a infinito.

Como consecuencia de lo discutido aquí, el funcional del PCPP debe incluir un quinto costo que se denominará "Costo Objetivo". Si se denomina target(x) al 'inventario objetivo' del producto 'x' y 'FDeficit' al factor que penaliza no alcanzar dicho inventario, entonces:

$$C.\,Objetivo = \sum_{x \in Tot_{Productos}} \left(\max(target(x) - Stock(x, horizonte), 0) + Ruptura(x, horizonte)\right) * FDeficit \qquad (7)$$

Cabe realizar dos observaciones sobre esta fórmula. En primer lugar, se utiliza la función 'máximo' para representar la asimetría en la relación entre el 'inventario objetivo' y el *stock* final. Es decir, en caso de no alcanzar el objetivo, supone un costo. Sin embargo, en caso de sobrepasarlo, no supone un beneficio. En segundo lugar, se incluye el valor de "Ruptura" al finalizar el horizonte de estudio, es decir, la demanda no satisfecha acumulada. Tiene sentido agregar este término ya que puede ser considerado como un stock negativo. Conceptualmente, agregar este término asigna el mismo 'Costo Objetivo' a las siguientes dos situaciones:

i) Un 'inventario objetivo' de 10 unidades, un stock final de 10 unidades y una demanda no satisfecha de 10 unidades.

ii) Un 'inventario objetivo' de 10 unidades, un stock final de 0 unidades y una demanda no satisfecha de 0 unidades.

Si no se agrega el término de 'Ruptura' al cálculo del 'Costo Objetivo', entonces la situación (i) sería preferible sobre la situación (ii), lo que no refleja la realidad.

A continuación, se describirán las funciones de entrada necesarias para introducir la información de las modificaciones hasta aquí descriptas del PCPP en relación al PAPP.

- *Lead-time:* Es una función de una variable LEADTIME(i) que indica el tiempo necesario de transporte entre la etapa productiva 'i' y la 'i+1'. Para la última etapa, esta función representa el tiempo entre la etapa productiva y el almacén de producto terminado.
- *Setup:* Es una función vectorial de tres variables SETUP(i,x,y) que devuelve el tiempo de *setup* y el costo por cada 'utem' para pasar de fabricar el producto 'x' al 'y' en la máquina 'i'.
- *Inventario Objetivo:* Se debe introducir una función de una variable TARGET(x) que indica el inventario objetivo del producto 'x' y un parámetro FDEFICIT que indica el factor de penalización por no alcanzar dicho inventario.

En el planteo más general, donde se enfrente un PCPP multi-época, también será necesario introducir:

- *Épocas:* 'Total Épocas' es un parámetro que indica la cantidad total de épocas en las que será dividido el horizonte. La función DURACION(i) indica la duración en 'utem' de la época 'i'. Es evidente que se debe cumplir la siguiente relación:

$$\sum_{i=1}^{Total\ Epocas} DURACION(i) = Horizonte$$

En este punto, se puede consolidar la nueva información de entrada con la del PAPP ya estudiado para realizar el siguiente cuadro (Tabla 4.1) que explicita toda la información necesaria para definir inequívocamente un PCPP multi-época, el problema más completo que se estudiará en el presente trabajo:

Categoría	Nombre	Naturaleza	Dominio	Imagen	Unidad	Tipo de dato
Productos	Productos	Conjunto	-	-	-	String
Máquinas	Total Etapas	Parámetro	-	-	[etapas]	Entero
	N° Máquinas	Función	(i en 1 .. Total Etapas)	Máquinas en etapa i	[máquinas]	Entero
Épocas	Total Épocas	Parámetro	-	-	[épocas]	Entero
	DURACION	Función	(i en 1 .. Total Épocas)	Duración de la época i	[utem]	Entero
Horizonte Temporal	Horizonte	Parámetro	-	-	[utem]	Entero
Lista de Materiales	BOM	Función	(x en Tot_Productos, y en Tot_Productos)	Cantidad de x para producir una unidad de y	[ust]	Float
Demanda	DEMANDA	Función	(x en Tot_Productos, i en 1 .. Total Épocas)	Demanda de x durante la época i	[ust/utem]	Float
Capacidad	CAPACIDAD	Función	(x en Tot_Máquinas, i en 1 .. Total Épocas)	Capacidad temporal de x durante la época i	[utem]	Float
Definición de Materiales	MATERIALDEF	Función	(x en Tot_Productos)	1) Precio de venta de x	[$/ust]	Float
				2) Costo de almacenamiento de x	[$/(ust*utem)]	Float
				3) Costo de ruptura de x	[$/(ust*utem)]	Float
Producción de Materiales	MATERIALPROD	Función	(x en Tot_Máquinas, y en Tot_Productos)	1) Ritmo de producción	[ust/utem]	Float
				2) Costo de Producción	[$/ust]	Float
Lead-Time	LEADTIME	Función	(i en 1 .. Total Etapas)	lead-time entre la etapa i y la etapa i+1	[utem]	Entero
Setup	SETUP	Función	(i en Tot_Máquinas, x en Tot_Productos, y en Tot_Productos)	1) Tiempo de setup en la máquina i para pasar de producir x a y	[utem]	Entero
				2) Costo por 'utem' de setup en la máquina i para pasar de producir x a y	[$/utem]	Float
Inventario Objetivo	TARGET	Función	(x en Tot_Productos)	Inventario objetivo de x	[ust]	Float
	FDEFICIT	Parámetro	-	-	[$/ust]	Float

Tabla 4.1: Resumen de información de entrada para la definición de un PCPP multi-época

Una vez definido correctamente el problema, se puede pasar a desarrollar su solución. Dicho desarrollo se explicita en la siguiente sección.

4.3 Solución del PCPP multi-época

Al igual que la resolución del PAPP, ésta será encarada como un proceso de cuatro etapas:

i) Generar nueva información útil a partir de la información de entrada.

ii) Generar la estructura de las variables de decisión.

iii) Expresar el funcional en términos de las variables de entrada y de decisión.

iv) Expresar las restricciones en términos de las variables de entrada y de decisión.

Siguiendo la metodología adoptada en la sección precedente, se explicarán únicamente los aspectos diferenciales de la resolución del PCPP. Sin embargo, a

la hora se condensar la información en cuadros, se agregará la información desarrollada en la resolución del PAPP para facilitar su comprensión y acceso.

i) Generar nueva información útil a partir de la información de entrada

Es necesario recordar que en esta primera etapa de la resolución se generan algunas funciones que contienen información ya existente en los datos de entrada pero de difícil acceso. De esta forma, se lleva la información a un formato sencillo de manejar para el algoritmo de resolución.

Por consiguiente, se generan las siguientes estructuras:

- *EtapaMaterial*: Esta es una función ETAPAMATERIAL(x) que depende de una variable e indica la etapa productiva en la que se fabrica el producto 'x'. Este dato es de especial necesidad a la hora de calcular el tiempo que tarda un producto cualquiera desde su fabricación hasta arribar al almacén intermedio. Este tiempo es el *lead-time* entre las etapas, haciendo que sea imprescindible relacionar cada producto con una etapa productiva. Notar que en la información de entrada esta relación se presenta de manera indirecta ya que se relaciona cada producto con una o más máquinas que lo pueden producir y luego éstas se relacionan con una etapa productiva. La función ETAPAMATERIAL(x) elimina la 'máquina' como intermediario en esta relación.

- *Comienzo*: Es una función de una variable COMIENZO(i) que almacena la primera 'utem' de la época 'i'. Esta información permite asignar cada 'utem' dentro de RHorizonte a su época correspondiente. Notar que COMIENZO(1) = 1 para todo problema y COMIENZO (N)= Horizonte+1 donde 'N' es el número total de épocas. Se calcula de la siguiente manera recursiva:

$$COMIENZO(1) = 1$$

$$COMIENZO(i) = COMIENZO(i-1) + DURACION(i-1)$$

$$\forall\, i = 2 \ldots Total\ Épocas$$

- *PosSet*: Es el análogo a *PosProd* pero para el *setup* de las máquinas. Se intenta explotar el esparcimiento del problema ya que, como es evidente, no todas las máquinas pueden realizar el *setup* para cambiar su producción entre dos productos cualesquiera. Por el contrario, deben ser productos que esa máquina pueda producir. Por consiguiente, se genera un conjunto que

guarde los posibles *setups* para cada 'utem'. De esta forma, se puede asegurar que:

$$card(PosSet) < card(Tot_{Maquinas})*card(Tot_{Productos})^2*card(RHorizonte)$$

Como consecuencia, se genera un ahorro en la memoria y la velocidad de procesamiento. A continuación, se define formalmente el conjunto generado:

$$PosSet = \{< i,j,l,k > / < i,j,l > \ \varepsilon \ Dom(SETUP) \ \& k \ \epsilon \ RHorizonte\}$$

En la tabla 4.2 se resumen las estructuras generadas, incluyendo aquellas desarrolladas en la resolución del PAPP.

Nombre	Contenido	Tipo de dato
RHorizonte	Todas las utem desde 1 hasta Horizonte	Entero
PosProd	(i en Tot_Maquinas, j en Tot_Productos, k in RHorizonte)	(String, String, Entero)
ETAPAMATERIAL(x)	Etapa productiva en la que se fabrica el material 'x'	Entero
COMIENZO(i)	Primera 'utem' de la época 'i'	Entero
PosSet	(i en Tot_Maquinas, j en Tot_Productos, l en Tot_Productos, k in RHorizonte)	(String, String, String,Entero)

Tabla 4.2: Estructuras generadas a partir de la información de entrada en el PCPP

ii) Generar la estructura de las variables de decisión

El PCPP requiere de una formulación más compleja de las restricciones, lo que redunda en un mayor número de variables de decisión a manejar. Las variables de decisión del PCPP incluirán todas aquellas del PAPP más algunas adicionales. Estas últimas se detallan a continuación:

- *MachSetup (<i,j,l,k> ϵ PosSet):* Variable binaria que indica si durante la 'utem k' la máquina 'i' realizó *setup* para pasar del producto 'j' al producto 'l'.
- *MachProd(<i,j,k> ϵ PosProd):* Variable binaria que indica si durante la 'utem k' la máquina 'i' estuvo produciendo el material 'j'. Notar que la definición en el PCPP es sensiblemente distinta de la definición en el PAPP. Antes era una variable continua que indicaba la proporción de la 'utem' en que se producía un material. Ahora, indica si se produjo o no un material durante una 'utem' dada. La proporción en que se produjo, está dada por la variable *'Sobra'* explicada a continuación.
- *Sobra(<i,j,k> ϵ PosProd):* Variable continua en el intervalo [0,1] que indica la proporción de la 'utem k' en la que la máquina 'i' estaba preparada para producir 'j' pero no lo hizo. Por ejemplo, si se supone que durante la 'utem'

3, la máquina “Maq11” estaba *‘seteada’* para producir el “AA” pero sólo produjo dicho material durante el 70% del tiempo de la ‘utem’ y el resto permaneció inactiva. En este caso, *MachProd*(<”Maq11”, “AA”, 3>) = 1 y *Sobra*(<”Maq11”, “AA”, 3>) = 0.3. Este planteo que puede resultar un tanto intrincado a primera vista, resulta algorítmicamente conveniente como se presentará mas adelante.

- *Disponible(i,j):* Nuevas ‘ust’ disponibles del material ‘i’ durante la ‘utem j’. Esta función es necesaria para introducir el efecto de retardo que existe entre que una unidad se produce hasta que puede utilizarse como producto semi elaborado o para su venta. Este retardo depende del *lead-time* de la etapa en que se produjo. Por ejemplo, se supondrá que durante la ‘utem 5’ se produjeron 10 ‘ust’ del material “BB” y que su *lead-time* es de 2 ‘utem’. En este caso, *Produ*(“BB”, 5)=10 y *Disponible*(“BB”,7)=10.
- *SumaParcialSetup(<i,j,l,k> ϵ PosSet):* Variable binaria que indica si en la ‘utem k’ la máquina ‘i’ está habilitada para realizar el cambio de la producción del producto ‘j’ al ‘l’. Obviamente, esta función está íntimamente relacionada con *MachSetup*. Por ejemplo, se supondrá que la máquina “Maq22” requiere de 2 ‘utem’ de *setup* para pasar de producir el producto “CC” al “DD”. Por lo tanto, si MachProd(<”Maq22”, “CC”, 5>) = MachSetup(<”Maq22”, “CC”, “DD”, 6>) = MachSetup(<”Maq22”, “CC”, “DD”, 7>) = 1, entonces, la máquina cumple todos los requisitos para realizar el cambio de producción, es decir, SumaParcialSetup(<”Maq22”, “CC”, “DD”, 8>) = 1.
- CSetup: Costo total de *Setup*.
- CObjetivo: Costo total por déficit en el inventario objetivo.

En la tabla 4.3 se condensa toda la información de las variables de decisión del PCPP.

Nombre	Dominio	Tipo de dato
Stock	(i en Tot_Productos, j en Rhorizonte)	Float
Ventas	(i en Tot_Productos, j en Rhorizonte)	Float
Ruptura	(i en Tot_Productos, j en Rhorizonte)	Float
Produ	(i en Tot_Productos, j en Rhorizonte)	Float
Uso	(i en Tot_Productos, j en Rhorizonte)	Float
MachProd	<i,j,k> en PosProd	Binaria
CAlmac	-	Float
CRuptura	-	Float
CProdu	-	Float
Ingreso	-	Float
MachSetup	<i,j,l,k> en PosSet	Binaria
Sobra	<i,j,k> en PosProd	Float
Disponible	(i en Tot_Productos, j en Rhorizonte)	Float
SumaParcialSetup	<i,j,l,k> en PosSet	Binaria
CSetup	-	Float
CObjetivo	-	Float

Tabla 4.3: Resumen de las variables de decisión del PCPP

Notar que parte de las variables de decisión no son de naturaleza continua, lo que implica que los algoritmos de resolución continua hasta aquí mencionados no serán de utilidad a la hora de resolver este problema. Por el contrario, serán necesarios algoritmos de programación entera mixta como el *'Branch and Bound'* y el *'Branch and Cut'* que serán desarrollados más adelante.

iii) Expresar el funcional en términos de las variables de entrada y de decisión

Al igual que en el PAPP, una vez definidas las variables de decisión, este paso es prácticamente trivial. Intentando maximizar el beneficio, se define:

$$Maximizar \quad Beneficio = Ingreso - CProdu - CAlmac - CRuptura - CSetup - CObjetivo$$

iv) Expresar las restricciones en términos de las variables de entrada y de decisión

En este caso se explicitarán todas las restricciones, incluso aquellas que ya existían en el PAPP. La idea es presentar la resolución en esta sección como completa, independiente de lo desarrollado en secciones anteriores. Además, la mayor parte de las restricciones sufre algún tipo de modificación, con lo cual el material redundante no será excesivo. Las restricciones serán numeradas

precedidas por el prefijo “RR” para distinguirlas de las enumeradas en la sección del PAPP.

Las restricciones se agruparán para facilitar su comprensión.

- *Imagen Acotada:* Las variables de decisión definidas como binarias poseen su imagen acotada a dos valores por naturaleza. Sin embargo, las variables de decisión continuas no pueden tomar cualquier valor porque representan variables físicas dadas. Por lo tanto, es necesario acotar los posibles valores de las variables de decisión:

  RR1) $Stock(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR2) $Ventas(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR3) $Ruptura(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR4) $Produ(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR5) $Uso(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR6) $Disponible(i,j) \geq 0 \; \forall \; i \in Tot_{Productos}, j \in RHorizonte$

  RR7) $0 \leq Sobra(< i,j,k >) \leq 1 \quad \forall < i,j,k > \in PosProd$

  RR8) $CAlmac \geq 0$

  RR9) $CProdu \geq 0$

  RR10) $CRuptura \geq 0$

  RR11) $CSetup \geq 0$

  RR12) $CObjetivo \geq 0$

  RR13) $Ingreso \geq 0$

- *Continuidad del Stock:* El planteo aquí es similar al del PAPP con la salvedad que el inventario no depende de lo producido sino de lo disponible. Esto se debe a que el transporte entre etapas productivas ya no lleva tiempo nulo. Escribiendo esto formalmente:

  RR14)

  $$Stock(i,1) = Disponible(i,1) - Ventas(i,1) - Uso(i,1) \qquad \forall\, i \in Tot_{Productos}$$

  RR15)

  $$Stock(i,j) = Disponible(i,j-1) + Produ(i,j) - Ventas(i,j) - Uso(i,j)$$
  $$\forall\, i \in Tot_{Productos}, j \in RHorizonte : j > 1$$

- *Continuidad de la demanda:* Esta restricción debe reflejar la naturaleza acumulativa de la demanda no satisfecha. Además, es necesario incluir el hecho de que la demanda depende de cada época. Escrito formalmente:

  RR16) $ruptura(i,1) + ventas(i,1) = DEMANDA(i,1) \quad \forall\, i \in Tot_{Productos}$

  RR17) $ruptura(i,j) + ventas(i,j) = DEMANDA(i,e) + ruptura(i,j-1)$

- *Usos como Semi Elaborado:* Con la producción de todos los productos excepto aquellos en la primera etapa productiva, se consumen otros materiales como semi elaborados. Esta relación debe formalizarse en una restricción como la siguiente:

  RR18) $Uso(i,j) = \sum_{h \in T(i)} Produ(h,j) * BOM(i,h)$

  Donde T(i) es el conjunto de productos que requieren del semi elaborado 'i' para su producción.

- *Continuidad de la Producción:* Se debe crear una restricción que vincule el tiempo de trabajo de las máquinas, es decir, 'MachProd' y 'Sobra' con la producción obtenida, o sea, 'Produ'. Dicha restricción se explicita a continuación:

RR19)

$$Produ(i,k) = \sum_{jeTot_{Maquinas}} \left(MachProd(<i,j,k>) - Sobra(<i,j,k>)\right) * MATERIALPROD_1(j,i)$$

$$\forall\ i \epsilon\ Tot_{Productos}, k\ \epsilon\ RHorizonte$$

- *Relación entre Disponibilidad y Producción*: Ya se explicó con anterioridad que los productos fabricados no pueden ser utilizados sino hasta que transcurra cierto tiempo de transporte interno. Esta relación se presenta a continuación:

RR20)

$$Disponible(i,j) = Produ\left(i, j - LEADTIME\left(ETAPAMATERIAL(i)\right)\right)\ \forall\ i\ \epsilon\ Tot_{Productos}, j\ \epsilon\ RHorizonte$$

Notar que de esta forma se discretizó la definición continua de *lead-time* que se había dado en la descripción del PCPP.

- *Relación entre Sobra y MachProd*: De las restricciones RR4 y RR18 se desprende el hecho de que *Sobra* debe ser menor o igual que *MachProd*, ya que de lo contrario, *Produ* sería negativo para algún índice y esto violaría la restricción RR4. De todas formas, como se comprenderá al analizar los algoritmos de resolución para problemas de programación entera mixta, conviene explicitar dicha relación para acotar el universo de soluciones factibles. Por lo tanto:

RR21) $Sobra(<i,j,k>) \leq MachProd(<i,j,k>) \quad \forall <i,j,k> \epsilon\ PosProd$

- *Evitar múltiples producciones y Setups:* Una simplificación que se introduce en este punto es que para una 'utem' dada, una máquina pudo haber estado en *setup* o producido como máximo un producto. Esto no significa que la máquina debe producir durante toda la 'utem', pero implica que una máquina dada no puede producir dos productos distintos en la misma 'utem'. Formalmente se escribe:

RR22)

$$\sum_{<i,j,k>\in PosProd} MachProd(<i,j,k>) + \sum_{<i,j,l,k>\in PosSet} MachSetup(<i,j,l,k>) = 1$$

$$\forall\, i \in Tot_{Maquinas}, k \in RHorizonte$$

Notar que en caso en que la máquina permanezca inactiva durante toda la 'utem', entonces MachProd valdrá 1 para el producto en que esté *seteada* la máquina pero Sobra también valdrá uno. Por lo tanto, la producción será nula.

- *Capacidad Temporal de las Máquinas:* En este caso, la capacidad temporal de cada máquina también depende de la época que se esté tratando. Además, las 'utem' no sólo se consumen cuando la máquina produce sino también cuando la máquina se encuentra en *setup*. De este modo:

RR23)

$$\sum_{j\in Tot_{Productos}} \sum_{k=COMIENZO(e)}^{COMIENZO(e+1)-1} \left(\left(MachProd(<i,j,k>) - Sobra(<i,j,k>)\right) + \sum_{\substack{l\in Tot_{Productos} \\ l\neq j}} MachSetup(<i,j,l,k>) \right) \leq CAPACIDAD(i,e)$$

$$\forall\, i \in Tot_{Maquinas}, e \in 1..Total\ Épocas$$

Sin embargo, si se introduce la restricción RR21 dentro de la RR22, se simplifica notablemente esta última, demostrando el ahorro en cálculo que genera utilizar la variable *Sobra.*

RR23´)

$$DURACION(e) - \sum_{j \in Tot_{Productos}} \sum_{k=COMIENZO(e)}^{COMIENZO(e+1)-1} Sobra(< i,j,k >) \leq CAPACIDAD(i,e)$$
$$\forall\, i \in Tot_{Maquinas}, e \in 1..Total\ Épocas$$

- *Restricciones sobre SumaParcialSetup:* Como se explicó oportunamente, esta variable de decisión detecta aquellos instantes en los que una máquina tiene permitido realizar un cambio en el producto que está fabricando. Para que un cambio entre un producto 'x' y un producto 'y' sea válido, es necesario que se haya estado produciendo el 'x' y luego haya una cantidad de 'utem' de *setup* intermedias necesarias para preparar la máquina. Sólo en ese caso se puede comenzar a fabricar el producto 'y'. Escribiendo esto formalmente:

  RR24)

$$SumaParcialSetup(< i,j,l,k >) = 0\ si\ k \leq SET$$
$$SumaParcialSetup(< i,j,l,k >) \leq MachProd(< i,j,k - SET - 1 >)$$
$$SumaParcialSetup(< i,j,l,k >) \leq MachSetup(< i,j,l,k - s >)\ con\ s = 1..SET$$
$$Donde\ SET = SETUP_1(i,j,l)\ \ \forall < i,j,l,k > \in PosSet$$

  Esta restricción que aparentemente es complicada, no lo es tanto si se interpreta de manera intuitiva. El primer renglón, comunica que es imposible pasar de producir 'j' a producir 'l' si el tiempo transcurrido no es al menos el tiempo necesario para realizar el *setup* entre estos dos productos. El segundo renglón, expresa que es imposible pasar de producir 'j' a producir 'l' si antes de comenzar el *setup* la máquina no estaba *seteada* para producir 'j'. Por último, el tercer renglón dice que es imposible realizar el cambio de producción si la máquina no estuvo realizando *setup* durante las 'utems' requeridas de acuerdo a la terna (máquina, producto que se producía, producto que se va a producir).

- *Habilitación para el cambio de producción:* Finalmente, se debe relacionar la función SumaParcialSetup con MachProd, comunicando cuándo se pueden realizar los cambios de producción. Como regla general, una máquina puede producir el producto 'x' durante la 'utem k' si estaba habilitada para producirlo durante la 'utem k-1' o si está habilitada para comenzar a producirlo durante la 'utem k'. Formalmente:

RR25)

$$MachProd(< i,j,k >) \leq MachProd(< i,j,k-1 >) + \sum_{<i,l,j,k> \in PosSet} SumaParcialSetup(< i,l,j,k >)$$

$$\forall < i,j,k > \in PosProd : k > 1$$

Notar que la restricción RR24 sólo se aplica para los k>1. Es decir, para la primera 'utem' se puede comenzar a fabricar cualquier producto. Esto es análogo a suponer que al comienzo del período de estudio, todas las máquinas se encuentran vacías.

- *Costo de Almacenamiento:* El costo de almacenamiento se calcula de manera análoga al PAPP:

RR26)

$$\sum_{i \in Tot_{Productos}} \left(\sum_{k=1}^{Horizonte-1} Stock(i,k) * MATERIALDEF_2(i) \right) + Stock(i,Horizonte) * 0{,}5 * MATERIALDEF_2(i)$$

Notar que el stock al comienzo de la primera 'utem' es nulo, por eso no aparece en la restricción.

- *Costo de Ruptura:* El costo de ruptura se calcula igual que en el PAPP:

RR27)

$$\sum_{i \in Tot_{Productos}} \left(\sum_{k=1}^{Horizonte-1} Ruptura(i,k) * MATERIALDEF_3(i) \right) + Ruptura(i,Horizonte) * 0{,}5 * MATERIALDEF_3(i)$$

- *Costo de Producción:* El cálculo del costo de producción difiera al de su análogo en el PAPP por la incorporación de la variable *Sobra.*

RR28)

$$\sum_{<i,j,k> \in PosProd} \left(MachProd(< i,j,k >) - Sobra(< i,j,k >) \right) * MATERIALPROD_1(i,j) * MATERIALPROD_2(i,j)$$

- *Costo de Setup:* El costo de *setup* se calcula simplemente como la suma de todas las 'utem' en las que se realizó *setup* multiplicadas por su costo unitario temporal.

RR29)

$$\sum_{<i,j,l,k>\epsilon PosSet} MachSetup(< i,j,l,k >) * SETUP_2(i,j,l)$$

- *Costo por 'Inventario Objetivo':* Cabe recordar que este es el costo en el que se incurre cuando no se cumplen con los 'inventarios objetivo' para los productos al finalizar el horizonte de estudio. Su cálculo se desprende de la definición discutida en la descripción del PCPP:

RR30)

$$C.Objetivo = \sum_{x\epsilon Tot_{Productos}} \left(\max(TARGET(x) - Stock(x,Horizonte), 0) + Ruptura(x,Horizonte)\right) * FDEFICIT$$

- *Ingreso:* El ingreso se calcula a partir de las ventas, igual que en el PAPP:

RR31)

$$\sum_{i\epsilon Tot_{Productos}} \sum_{k=1}^{Horizonte} Ventas(i,k) * MATERIALDEF_1(i)$$

Se puede contemplar que las restricciones del PCPP comparadas con las del PAPP no han crecido únicamente en número sino también en complejidad. La función SETUP, con tres variables de entrada, aumenta notablemente el requerimiento de cómputo en la resolución. Además, se utiliza una función no lineal ("Máximo") que, aún cuando los argumentos sean continuos, utiliza funciones lógicas para su cálculo que son representadas por variables binarias en cualquier motor de resolución computacional.

A continuación se explicarán los algoritmos de resolución de programación entera mixta y luego se aplicarán computacionalmente para resolver el PCPP.

4.4 Algoritmos de resolución para problemas enteros mixtos

Como se desprende de la descripción del PCPP, parte de las variables de decisión son de naturaleza binaria. Como consecuencia, los algoritmos hasta aquí descriptos (Simplex Primal, Simplex Dual y Barrera) carecen de utilidad.

En esta sección se explicarán brevemente dos algoritmos, íntimamente relacionados entre sí, que resuelven problemas de programación entera mixta: '*Branch and Bound'* y '*Branch and cut'*. La explicación no pretende ser exhaustiva desde el punto de vista matemático. Por el contrario, se intentan introducir algunos conceptos de forma intuitiva para luego poder ser utilizados a la hora de analizar la solución.

4.4.1 Branch and Bound

Primero, es necesario entender qué significa una relajación lineal de un problema entero. Una *relajación lineal* es simplemente el problema de optimización lineal que surge de quitar los requerimientos de discretización de todas las variables de decisión enteras (incluyendo las binarias). Es decir, se impone que todas las variables de decisión sean continuas.

El algoritmo *Branch and Bound* (B&B) comienza resolviendo la relajación lineal del problema original. Para esto, se utiliza cualquiera de los algoritmos para resolución de problemas continuos explicados hasta este punto. Una vez obtenida la solución de este problema, se toma alguna de las variables (llámese x) cuyo valor óptimo no es entero (por ejemplo, 3.3) pero que el problema original requiere que lo sea, y se ramifica el espacio de soluciones factibles en dos. Cada nuevo espacio es una relajación lineal del problema original pero con una nueva restricción. A un espacio se le exige que x<=3 y al otro se le exige que x>=4. De esta forma, se resuelven ambos problemas y se vuelve a ramificar según corresponda.

La figura 4.2 muestra la relajación lineal de un problema donde ambas variables de decisión (X e Y) requieren ser enteras en el problema original y se intenta maximizar el funcional Z(X,Y). La siguiente figura 4.3, muestra la partición del espacio de soluciones factibles para la primera ramificación del algoritmo. Notar que el óptimo de la relajación del problema original, al no ser una solución entera, queda fuera del espacio de soluciones factibles luego de la ramificación.

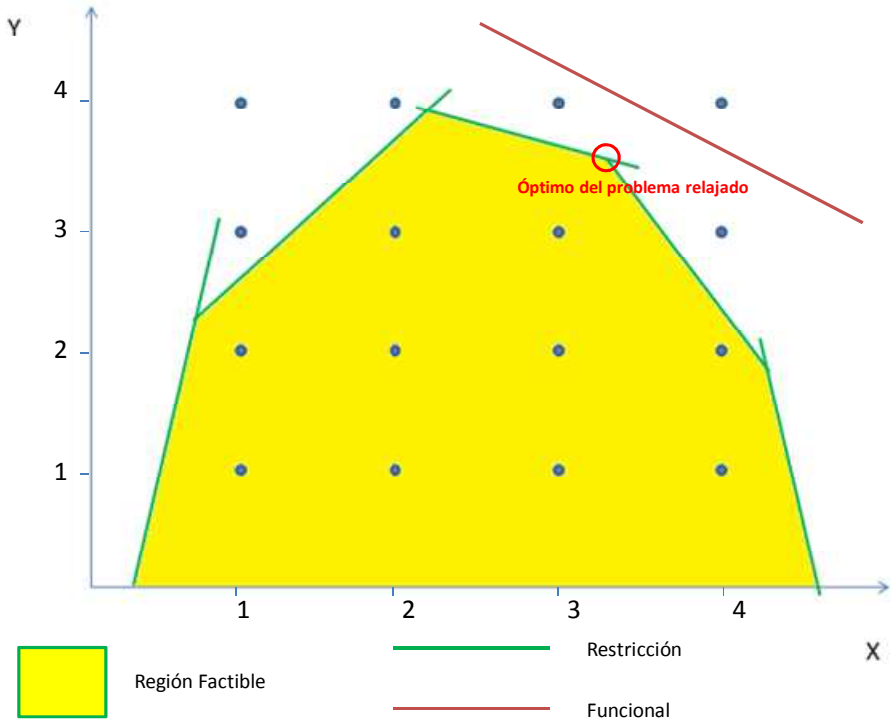

Figura 4.2: Relajación lineal del problema original

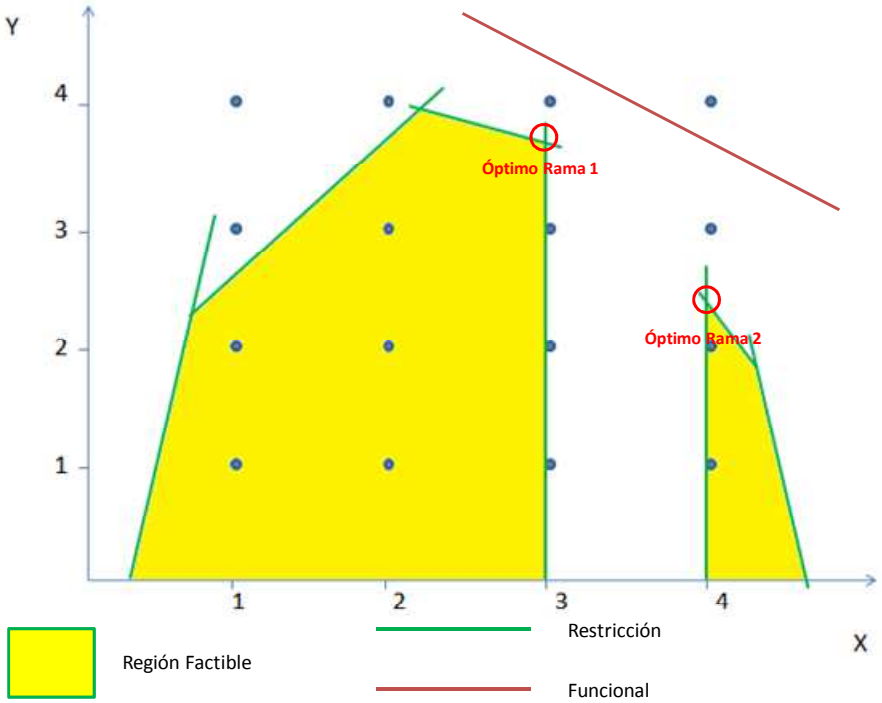

Figura 4.3: Partición del espacio de soluciones factibles para la primera ramificación

Como se puede observar en la figura precedente, ambos nuevos óptimos son no enteros en la variable Y. Por lo tanto, la ramificación debe continuar. El esquema de la figura 4.4 muestra las sucesivas ramificaciones mientras que sus soluciones fueron plasmadas en la figura 4.5. La comprensión de ambas figuras asegura un sólido entendimiento de la naturaleza básica del algoritmo.

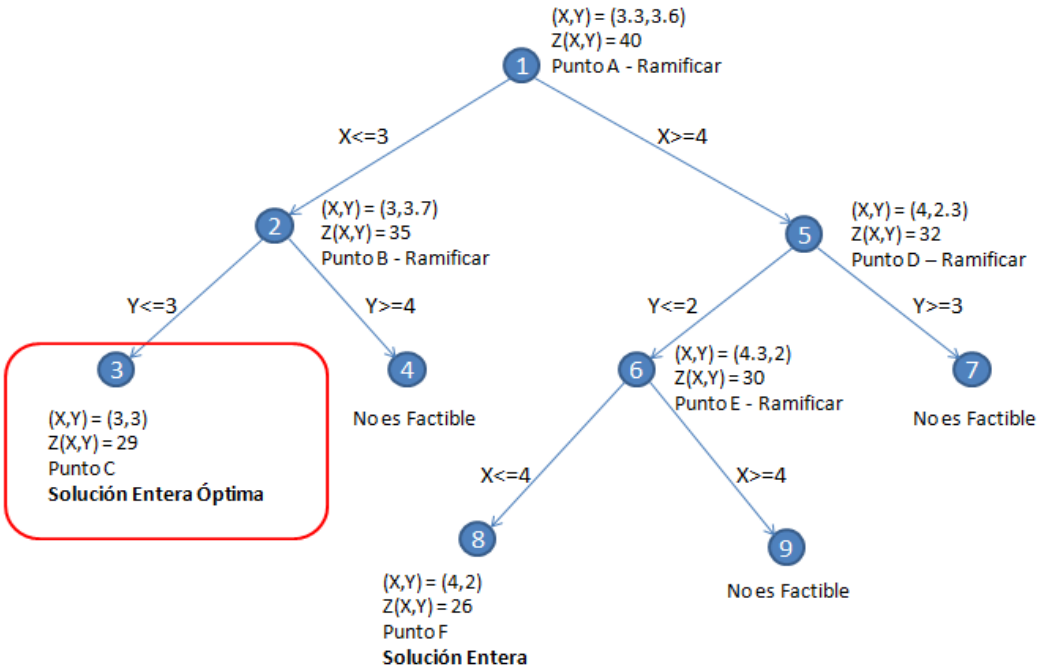

Figura 4.4: Sucesivas ramificaciones durante la resolución del problema

Es de interés observar que en el árbol desarrollado en la figura 4.4 todas las ramas terminan ya sea en una solución entera (la mayor de todas es la óptima ya que se está maximizando) o en una solución no factible. Sin embargo, no es éste siempre el caso. Es evidente que la solución a la relajación de un problema es siempre mayor igual (en caso de estar maximizando) a su solución entera. No es el objetivo demostrar esto matemáticamente, simplemente alcanza con la idea intuitiva de que la relajación del problema tiene menos restricciones por lo cual tiene un espacio de soluciones factibles más amplio.

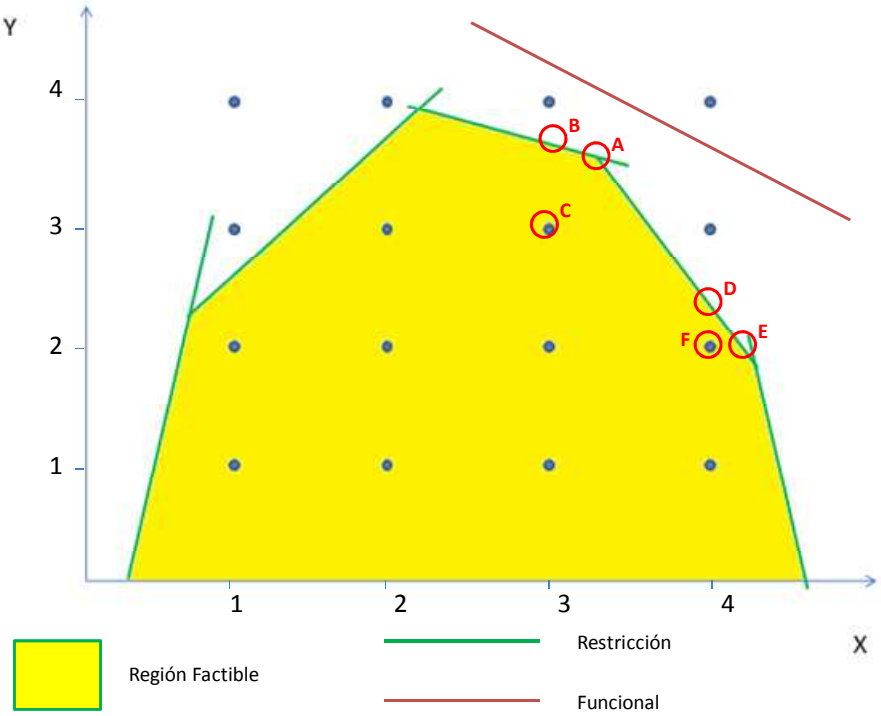

Figura 4.5: Ubicación de los sucesivos puntos óptimos a lo largo de la ejecución del algoritmo

Esto implica que su solución óptima debe ser mayor o igual a la solución óptima de cualquier sub espacio de soluciones factibles, incluyendo aquellos que surgen de discretizar las variables. Este hecho provee una pauta interesante para 'cortar' el árbol de ramificaciones: *si en cualquier rama del árbol se encontró una solución entera mejor que la relajación de otra rama, entonces no se debe seguir trabajando sobre ésta última ya que ninguna de sus posibles soluciones enteras puede ser óptima*.

En el ejemplo descripto, si Z en la rama 6 fuese menor o igual a 29 ("Rama 3"), no tiene sentido inspeccionar las ramas 8 y 9 ya que es imposible que el valor de su funcional sea superior a 29 y ya se ha encontrado una solución entera de ese valor.

Del análisis hasta aquí realizado, se desprenden algunas características del algoritmo B&B que vale la pena mencionar:

- El B&B supone una resolución sucesiva de varios problemas de optimización lineal continua. La cantidad de estos problemas a resolver es igual a las 'ramas' estudiadas del árbol. Si bien las ramas pueden cortarse antes de su desarrollo total, su forma de multiplicación es la explicación del

crecimiento exponencial del tiempo de resolución con la cantidad de variables enteras a considerar (ver sección precedente “Naturaleza de las unidades físicas”).

- En contraste con los algoritmos lineales, B&B tarda cierto tiempo en encontrar la solución óptima y luego tarda otro período de tiempo en demostrar que dicha solución es efectivamente la óptima. En el ejemplo, se supondrá que las ramas se fueron estudiando en el orden de su numeración. La solución óptima fue hallada al inspeccionar la tercera rama, pero el algoritmo no pudo asegurar su optimalidad hasta inspeccionar seis ramas más. Es decir, en este caso, B&B tardó más tiempo demostrando optimalidad que encontrando la solución óptima.

- El tiempo hasta hallar la solución óptima depende en gran medida del ‘camino’ seguido dentro del árbol. Esto hace que el tiempo de resolución, además de depender del tamaño del árbol, dependa de los valores de las variables que determinan el ‘camino’ a seguir. Esto lleva a una cantidad de estrategias que ayudan a determinar qué rama conviene seguir en cada bifurcación. Sin embargo, la profundidad del presente trabajo no justifica su desarrollo. Otra estrategia notable ofrecida por algunos software (por ejemplo, CPLEX) es una resolución en paralelo del árbol, en dónde cada procesador de la computadora sigue un ‘camino’ distinto dentro del árbol y se retroalimentan entre ellos para generar los cortes. Esto genera una mejora notable en el tiempo de resolución.

- Suponiendo un problema de maximización, la solución entera óptima está limitada inferiormente por la mejor solución entera hallada hasta el momento y superiormente por la mejor rama factible no bifurcada. Cuando estos dos límites concuerdan, se llegó a la solución buscada. La distancia porcentual entre estos dos límites se suele denominar *Gap* y es de gran interés al estudiar la performance del algoritmo. El límite inferior suele modificarse con saltos discretos cuando se halla una nueva solución entera mientras que el límite superior se reduce de manera casi constante para árboles de gran tamaño. Esto se grafica en la figura 4.6 a continuación. La línea verde indica la mejor solución entera hallada hasta el momento mientras que la roja indica la mejor rama factible no bifurcada hasta ese momento. Cuando ambas convergen, la optimización ha concluido. Notar que la solución óptima fue hallada en aproximadamente siete segundos,

pero se necesitaron treinta segundos más para probar su optimalidad. En problemas de gran tamaño, la solución entera converge asintóticamente al mejor nodo. Con lo cual, se suele fijar un *Gap* aceptable (por ejemplo del 5%) y se considera finalizado el algoritmo cuando el *Gap* alcanzado es menor o igual al aceptable. En otras palabras, un *Gap* del 5% significa que el óptimo global tiene un funcional *a lo sumo* 5% mejor que el hallado hasta el momento.

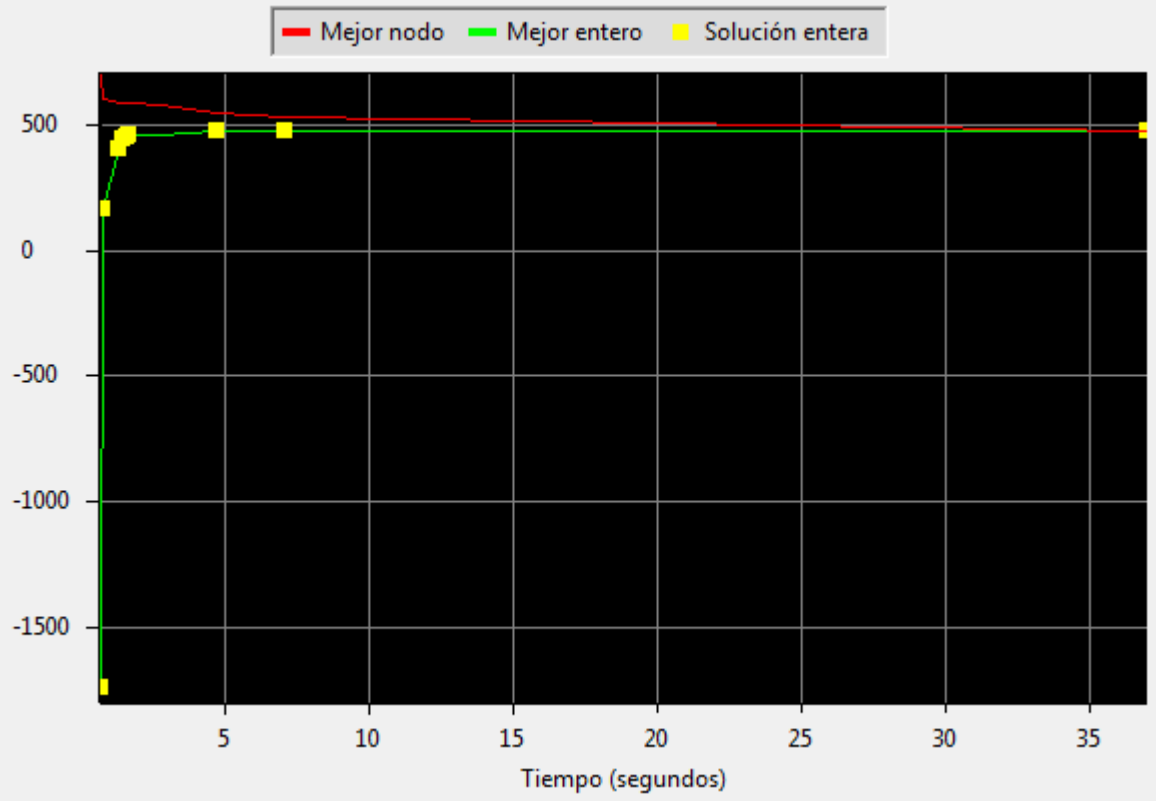

Figura 4.6: Progresión de la resolución de un problema de programación entera mixta

- El árbol a analizar depende fuertemente de las restricciones propuestas. Como consecuencia, la explicitación de *restricciones adicionales válidas* reduce radicalmente el tiempo necesario de resolución. Las *restricciones adicionales válidas* son restricciones que se imponen a la relajación del problema que reducen el espacio de soluciones factibles sin dejar excluida a ninguna solución entera que se incluía en la relajación original. Esta idea da origen al otro algoritmo a estudiar.

4.4.2 Branch and Cut

El algoritmo *Branch and Cut* (B&C) es un híbrido entre al B&B y la técnica de los *planos cortantes*. Una breve descripción del algoritmo aclarará esta idea.

El algoritmo comienza resolviendo una relajación del problema original (igual que el B&B) pero, si la solución no es entera, introduce una nueva restricción lineal que *corta* el espacio de soluciones factibles (de allí el nombre *planos cortantes*) de manera de no dejar ninguna posible solución entera afuera. Luego, vuelve a resolver el problema acotado. Si la solución continúa sin ser entera. Entonces aplica B&B, es decir, parte el espacio de soluciones factibles en dos espacios y aplica B&C a cada uno de los dos nuevos espacios generados.

La dinámica en la búsqueda de la solución (la creación y el recorrido del árbol, el desarrollo de los límites superiores e inferiores) es análoga al B&B. Sin embargo, es usual que los planos cortantes reduzcan el tamaño del árbol, acelerando la resolución.

Existe una gran variedad de tipos de cortes. Algunos de ellos se ajustan mejor a estructuras de datos que otros. Por ejemplo, hay algunos cortes funcionan cuando se tiene una serie de variables binarias donde sólo una porción de ellas puede tomar valores positivos. De todas formas, la explicación de los distintos tipos de cortes, excede el objetivo de la presente sección del trabajo.

A continuación, se ejemplifica un corte. Considerando la siguiente restricción donde x,y,z son variables binarias:

$$10*x + 12*y + 15*z < 20$$

Es evidente que a lo sumo una de las tres variables puede tomar valor 1, ya que si dos cualesquiera fuesen uno, la restricción no se cumpliría. De esta forma, se puede agregar el corte:

$$x + y + z <= 1$$

Este corte no deja fuera del espacio de soluciones factibles a ninguna solución entera que cumpliese la restricción original, pero deja fuera a soluciones fraccionarias como (x=1, y=0.6, z=0).

A continuación, en la figura 4.7 se grafica el primer paso de la aplicación de B&C al ejemplo estudiado en la sección del B&B. Notar que en este caso, el punto a partir del cual se ramifica es el A’ en lugar del A. Como consecuencia, el árbol que se desprende es de menor tamaño que el de antes resultando en una resolución más eficiente. No se presenta el desarrollo del árbol ya que el procedimiento es análogo al del B&B. Cabe mencionar que el CPLEX utiliza *Branch and Cut* como su algoritmo de resolución de problemas de programación entera mixta.

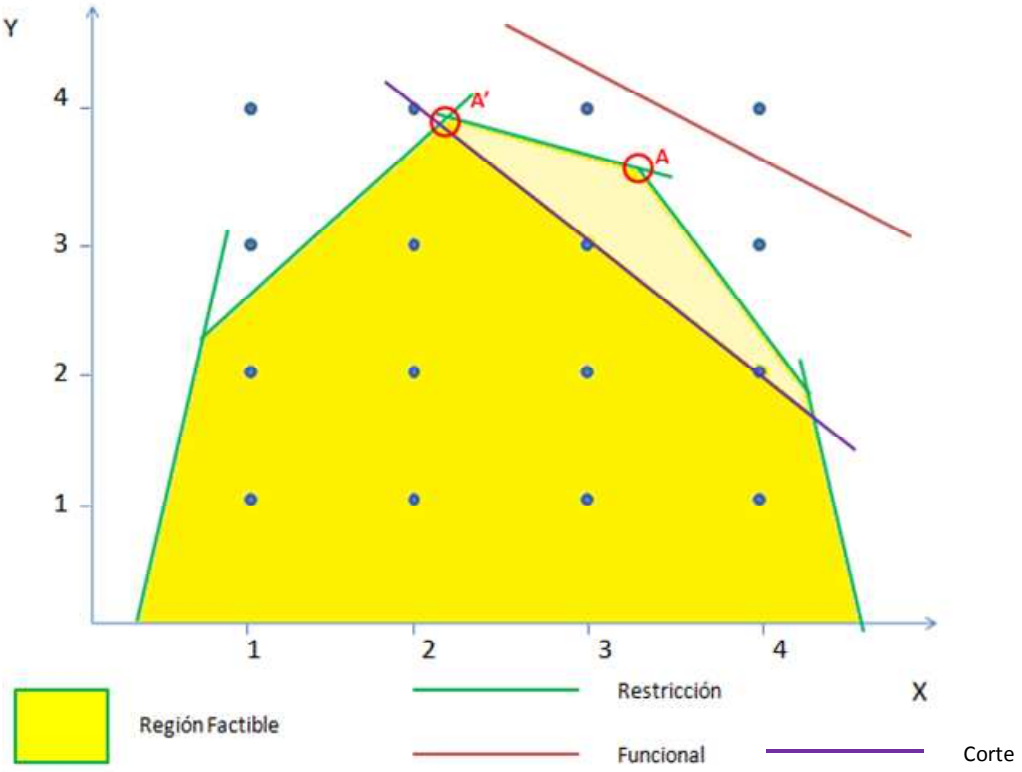

Figura 4.7: Primer paso en la aplicación del algoritmo branch and cut

4.5 Aplicación de la solución del PCPP multi-época

Al igual que en el PAPP, se aplicará la solución antes desarrollada utilizando el software de optimización matemática *IBM ILOG CPLEX.* Como resulta evidente, para que un software pueda resolver el PCPP debe soportar un algoritmo de resolución de problemas de programación entera mixta. En el caso del *CPLEX,* el algoritmo que utiliza es el *Branch and Cut* y se puede elegir qué método de resolución continua utilizar para la resolución de las relajaciones lineales en cada nodo. Siendo consistentes con el desarrollo hasta aquí, se elegirá el método de Barrera para los problemas continuos.

En el apéndice A (9.4), se puede encontrar un programa en *CPLEX* que aplica la resolución teórica desarrollada con anterioridad. Si bien algunos detalles de la sintaxis pueden no ser del todo comprensibles para alguien no acostumbrado al lenguaje, la mayor parte del programa resulta intuitivo. Más aún, se marcaron las treinta y un restricciones (RR1 hasta RR31) sobre el código para facilitar su correlación con la solución teórica.

Con el objetivo de comprender el alcance de dicha solución, se aplicó el algoritmo mencionado a un set de datos que puede ser consultado en el apéndice B (10.3). La configuración de la fábrica y de los productos es similar a la de los ejemplos anteriores. Sin embargo, se agregó toda la información necesaria para convertirlo en un PCPP (*lead-times*, *setups* e 'inventarios objetivo'). Más aún, el horizonte temporal se redujo a 28 'utem' con cuatro épocas de 7 'utem' cada una. Se intenta simular un mes donde cada semana tiene un comportamiento distinto. Por esta razón, a lo largo de este ejemplo, se llamará 'día' a cada 'utem' y 'semana' a cada 'época'.

A continuación se presenta, a modo de ejemplo, alguna información útil que devuelve el programa. Desde ya, la lista que aquí se presenta está lejos de estar completa. La información que se puede extraer es muy diversa y va a depender del interés particular del usuario a la hora de resolver el PCPP. Sin embargo, se comienza presentando el objetivo principal del programa, es decir, la programación de la producción. A partir de allí, se presenta otra información simplemente para ejemplificar que la resolución va más allá del programa de producción, sino que aporta información de vital importancia para la gerencia.

En contraste con el PAPP, la solución del PCPP posee **valor cuantitativo** para la empresa. Esto no significa que la solución sea una fotografía de la performance futura ya que pueden fallar algunos de los supuestos. Entre estos supuestos se encuentran el determinismo de la demanda, la capacidad horaria de las máquinas que se puede ver reducida por mantenimiento inesperado o por problemas con los operarios, etc. Sin embargo, los supuestos que asume el PCPP se encuentran dentro de parámetros físicos lógicos. Por lo tanto, aún cuando la información de entrada no se pueda aseverar, la resolución es un norte al cual apuntar. Desde luego, a la programación de la producción obtenida como solución del PCPP, se le podrían realizar pequeños cambios para ajustarlo a contingencias menores que surjan a lo largo del camino. De esta forma, se minimizaría el apartamiento del beneficio óptimo. En caso en que las contingencias sean mayores (por ejemplo, una máquina se averió durante todo el período de estudio), simplemente habría que modificar la información de entrada y volver a correr el programa.

Al igual que en el PAPP, se presentan cinco ejemplos de la información de salida:

a) *Programación de la Producción*

La principal utilidad de la herramienta desarrollada a lo largo del presente trabajo es obtener lo que debe hacer cada máquina en cada momento del horizonte de estudio. Por lo tanto, resulta lógico presentar en primer lugar un diagrama de Gantt

con la producción de cada máquina para cada instante. Los tiempos de producción se marcan en verde y los de *setup* en celeste en la figura 4.8.

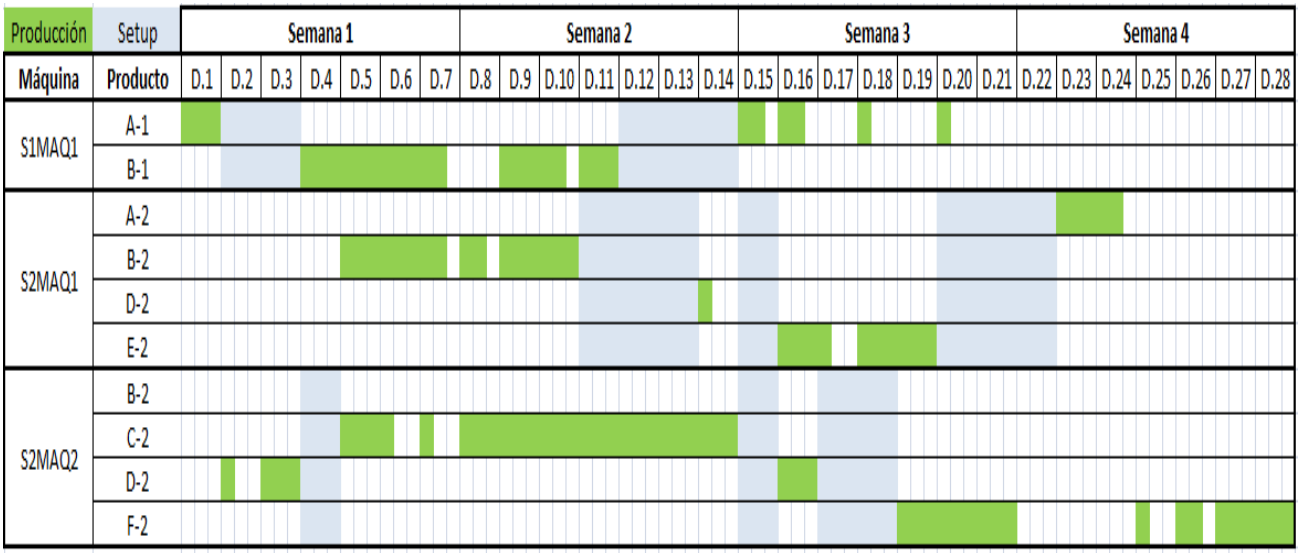

Figura 4.8: Diagrama de Gantt con la programación de producción solución del PCPP

Sobre el diagrama se pueden observar algunas características intrínsecas del problema que confirman su solidez. Por ejemplo, ninguna máquina de la segunda etapa (“S2MAQ1” y “S2MAQ2”) puede producir durante el primer día ya que los productos semi elaborados tardan un día en viajar desde la etapa anterior. Se puede fabricar “D-2” a partir del segundo día porque sólo requiere de “A-1” como insumo. Por el contrario, “B-2” sólo se puede fabricar a partir de que ambos insumos “A-1” y “B-1” arribaron al almacén intermedio de la segunda etapa. Esto se debe a que en la lista de materiales de “B-2” figuran ambos. Los tiempos de *setup* tienen perfecta correlación con los enunciados como información de entrada. Las capacidades de las máquinas se respetan. Por ejemplo, la máquina “S2MAQ2” durante la última semana trabaja sólo tres días en total, que es su máxima capacidad.

Como es evidente, de la solución del PCPP no sólo se puede extraer una instrucción a seguir para maximizar el beneficio, sino una herramienta de mejora hacia el futuro. Por ejemplo, si se examina el programa de la máquina “S1MAQ1” se puede ver que la capacidad limitante no influye demasiado sobre el beneficio ya que hacia el final del período tiene un alto componente de inactividad. Sin embargo, los tiempos de *setup* le impiden variar demasiado su producción. Por esto, es lógico suponer que un descenso (por ejemplo a 1 día) sobre sus tiempos de *setup* debería impactar directamente sobre el beneficio de la empresa. En este

caso, el usuario no tendría más que modificar la información de entrada y observar el beneficio en el nuevo escenario.

Obviamente, se pueden resolver PCPP de mayor envergadura en los que el diagrama de Gantt resultaría más complejo. De todas formas, el tiempo de resolución aumenta en gran medida con el tamaño del problema, cómo se estudiará más adelante.

b) *Destinos del ingreso*

Conocer el margen de ganancia sobre la facturación o el peso relativo de cada uno de los costos es esencial para una efectiva gerencia.

Por esto, resulta de vital importancia extraer información como la que se muestra en la figura 4.9. Más allá de los valores obtenidos para este ejemplo, que podrían ser considerados poco realistas, se deja en claro la utilidad de poder acceder a este tipo de información.

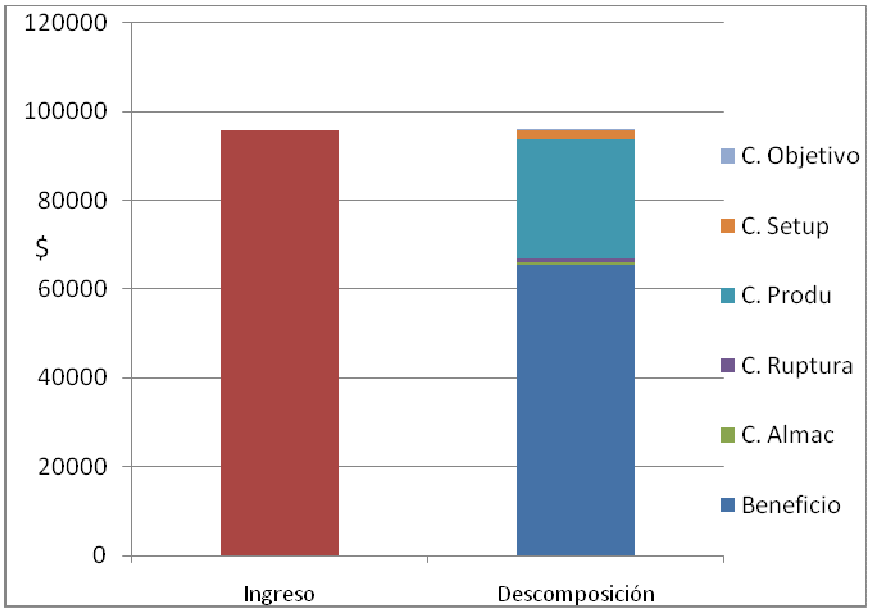

Figura 4.9: Descomposición del ingreso en beneficio y costos

En particular, en este ejemplo se puede observar un altísimo margen sobre el ingreso (aprox. 68%) y un costo productivo predominante sobre el resto (aprox. 88% del costo total). Sería natural buscar reducir el costo productivo si se busca un aumento en el margen, por ejemplo.

c) *Correlación entre Producción y Uso Interno*

Existe una relación lógica entre las distintas funciones que entran en juego en la optimización de la solución. Estas relaciones son las que se plasmaron en las restricciones desarrolladas en la resolución teórica. Se pueden generar gráficos que incluyan a las ventas, los inventarios, la producción, entre otras funciones. A modo de ejemplo, se grafica la relación entre el uso de los productos semi elaborados y la fabricación de los productos terminados. En otras palabras, la figura 4.9 grafica la restricción RR18.

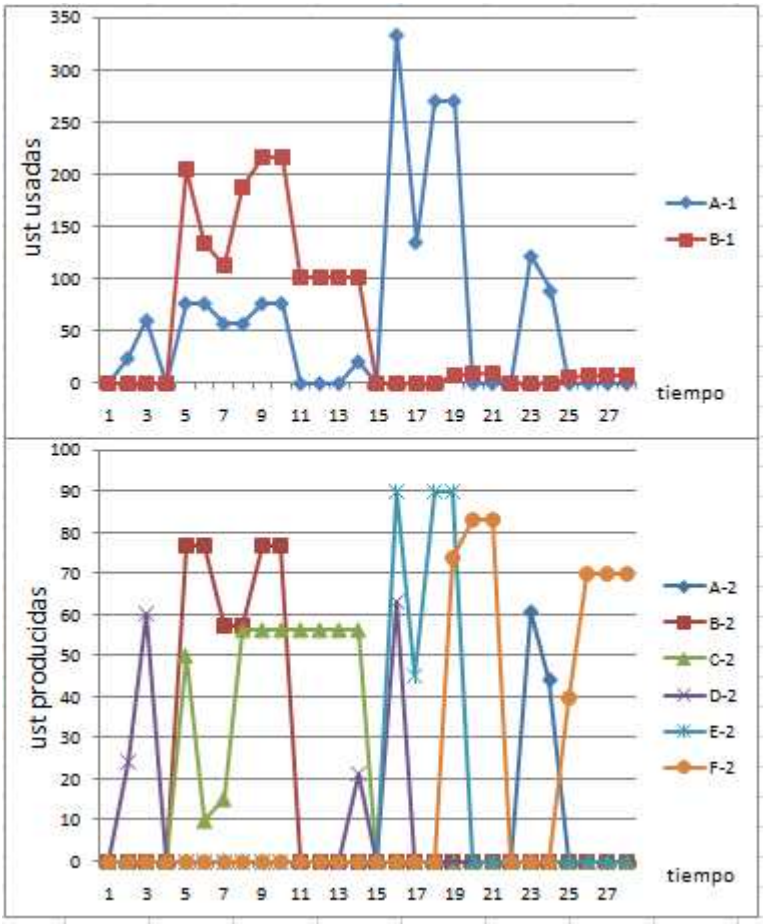

Figura 4.9: Representación gráfica de la relación entre productos semi elaborados consumidos y productos terminados fabricados

Se puede observar la marcada correlación entre los picos y los valles de ambos gráficos. Naturalmente, la magnitud de éstos se ve modificada por el factor BOM correspondiente.

d) *Capacidades y Uso de las máquinas*

Otra área de mejora se relaciona con las capacidades de cada máquina. Es de gran importancia detectar los "cuellos de botella" en el proceso para poder maximizar el impacto de cualquier mejora en el beneficio total. Por consiguiente, resulta útil tener acceso a información como la presentada en la figura 4.10.

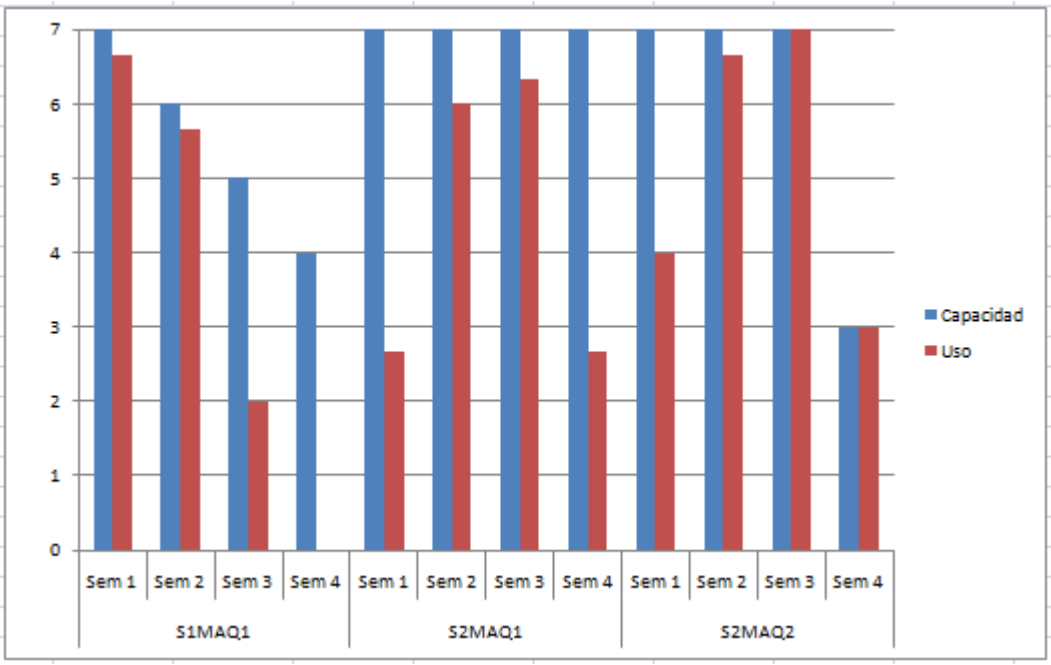

Figura 4.10: Capacidad y utilización semanal para cada máquina

De este gráfico se pueden extraer diversas conclusiones. Por ejemplo, se puede ver que la máquina "S1MAQ1" se encuentra más cargada durante las primeras semanas que durante las últimas. Por lo tanto, aún con la reducción de capacidad, se podría subalquilar esta máquina durante las últimas semanas para aumentar el beneficio. Por otro lado, la máquina "S2MAQ2" trabaja a toda capacidad hacia el final del horizonte de estudio, justo cuando tiene una marcada baja en capacidad. Si esta disminución se debiese a un mantenimiento preventivo, por ejemplo, se podría cambiar para la semana 1, impactando en un aumento en el beneficio. Más aún, se puede correr el programa con este escenario supuesto y observar que un cambio menor como el descripto genera un 35% de descenso en el costo de almacenamiento.

e) *Facturación semanal por producto*

Resulta de gran interés conocer qué producto es aquel que aporta mayor facturación cada semana. Esto facilita la generación de *rankings* entre los

productos que sirven a la hora de tomar decisiones. Para este ejemplo en particular, se puede observar la figura 4.11.

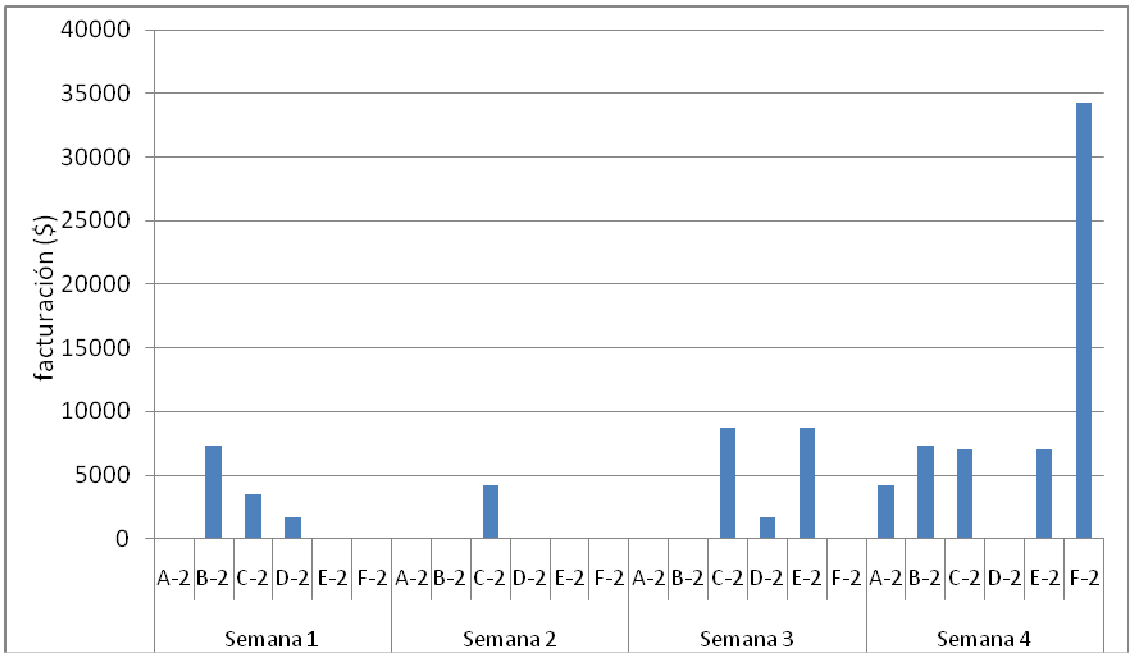

Figura 4.11: Facturación semanal de cada producto

Es sencillo observar que la mayor facturación proviene de las ventas de “F-2” durante la última semana de estudio. De esta forma, se podrían apuntar esfuerzos de marketing a impulsar la demanda de dicho producto en el resto de las semanas, por ejemplo.

4.6 Alcanzando el límite de la capacidad de cómputo

De la explicación del funcionamiento de los algoritmos de resolución para problemas mixtos, es decir que involucren variables de decisión enteras o binarias, queda claro que el tamaño del árbol a resolver no crece de forma lineal con la cantidad de variables. Por ejemplo, suponiendo que se cuenta con una cantidad ‘n’ de variables binarias, todas ellas independientes entre sí, entonces el peor escenario a resolver depende de 2^n. En el PCPP, el árbol a estudiar no crece exactamente de esta forma por dos razones. En primer lugar, es inherente al algoritmo que el árbol se ‘corte’ antes de desarrollarlo totalmente ya que hay ramas que no vale la pena continuar examinando como se explicó con anterioridad. En segundo lugar, las variables binarias no son independientes entre sí. Por ejemplo, para una ‘utem’ y máquina dada, sólo se puede estar fabricando a

lo sumo un producto. De esta forma, la cantidad de soluciones factibles se reduce notablemente.

Sin embargo, el crecimiento del árbol con la cantidad de variables binarias es pronunciado aún en el PCPP. Más aún, resulta evidente que la cantidad de variables binarias a considerar crece de manera lineal con el horizonte de estudio, pero de manera cuadrática con la cantidad de productos a considerar (por las variables relacionadas con el *setup*). Estos dos efectos combinados, hacen que la resolución exacta de un PCPP de dimensiones significativas sea imposible de computar.

A continuación, en la figura 4.12 se muestra el tiempo de resolución hasta un *Gap* del 5% (concepto introducido en la explicación del *Branch and Bound*) para el PCPP ejemplo variando la duración de cada una de las cuatro épocas.

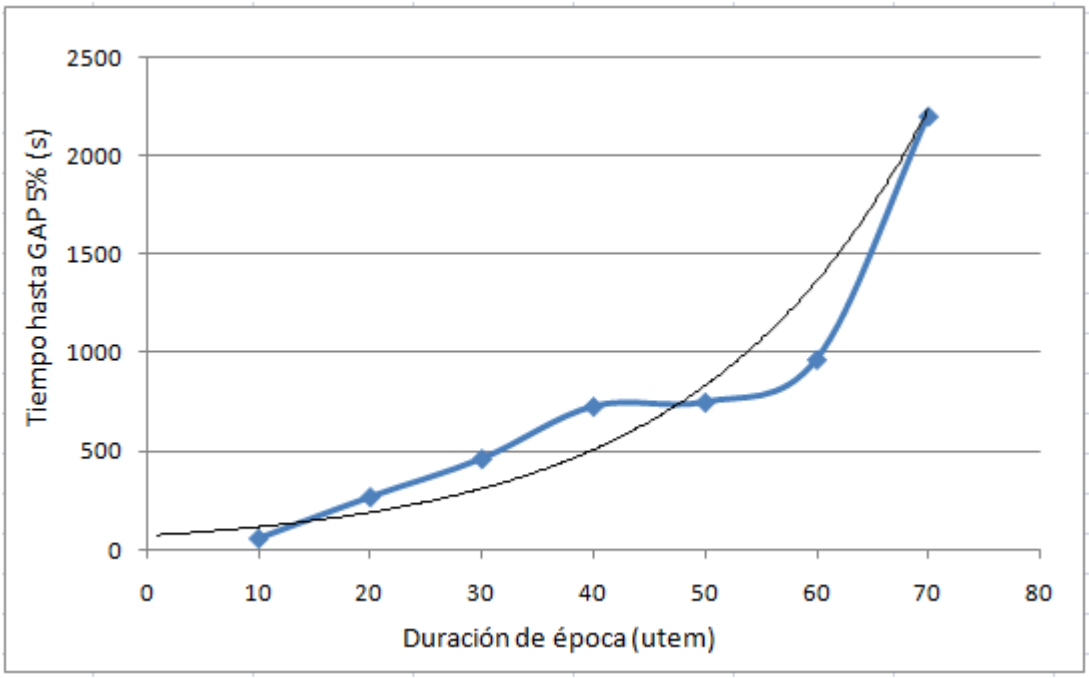

Figura 4.12: Relación entre el tiempo de resolución hasta un Gap del 5% y la duración de la época

En el gráfico se puede apreciar, además, la línea de tendencia exponencial. De la comparación de ambas gráficas, se puede inferir que el tiempo de resolución está siendo determinado por dos efectos simultáneos. Por un lado, el crecimiento ´cuasi´ exponencial del tamaño del árbol a resolver y, por otro lado, una variación aleatoria del camino tomado en la resolución. En otras palabras, el algoritmo puede demorar más en resolver un árbol de 500 nodos si comienza por el camino incorrecto que uno de 5000 si comienza por el camino indicado. Por esto, el tiempo varía aleatoriamente distanciándose de la tendencia exponencial.

Del gráfico y la discusión anterior se desprende que, hasta el momento, se ha desarrollado una herramienta teórica relativamente poderosa pero con un acotado alcance práctico ya que, cuando la dimensión del problema aumenta, el tiempo necesario para arribar a la solución óptima crece con mayor rapidez. Este hecho trunca parcialmente los objetivos de flexibilidad y alcance universal planteados en la concepción de la herramienta.

De todas maneras, la imposibilidad de alcanzar la solución exacta en un tiempo razonable no se debe a una debilidad de la herramienta, sino a una dificultad inherente a la resolución del PCPP. Categorizar la dificultad de resolver dicho problema no es el objetivo del presente trabajo, pero se puede demostrar que el problema general de programación de la producción es 'NP-hard' [Florian *et al.*, 1980].

Por esta razón, a lo largo del siguiente capítulo se presentan diferentes estrategias con un objetivo común: apartarse ligeramente de la rigurosidad teórica para obtener una mejora en el tiempo de procesamiento. En otras palabras, se resigna el deseo de encontrar el óptimo global para encontrar un óptimo local que sea una aproximación *'razonable'* del primero en un menor tiempo de resolución.

5 DEL ÓPTIMO LOCAL AL GLOBAL

5.1 Introducción

Como se mencionó con anterioridad, en el presente capítulo se describirán algunas estrategias desarrolladas con el objetivo de resolver el PCPP en menor tiempo que el que llevaría la resolución exacta explicada en el capítulo anterior. En otras palabras, se debe manejar un delicado equilibrio entre el apartamiento del óptimo global y la ganancia en tiempo de procesamiento.

La naturaleza de los algoritmos *'Branch and Cut'* y *'Branch and Bound'* permiten truncar la optimización con un límite temporal y obtener la mejor solución hallada hasta ese momento. Sabiendo esto, se podría plantear la siguiente alternativa: *correr la optimización completa del PCPP desarrollada en el capítulo anterior durante el máximo tiempo del que se disponga y utilizar la mejor solución hallada hasta ese momento*.

Si bien la idea detrás de este procedimiento no es errónea, sigue careciendo de sentido práctico ya que, en problemas de gran tamaño, el desarrollo del árbol es muy lento, lo que no asegura una solución razonable en el orden de los minutos o algunas horas. Por lo tanto, lo que se busca es alterar la formulación para encontrar "rápidamente" una solución razonable. En la figura 5.1 se esquematiza el beneficio de la mejor solución hallada en función del tiempo de resolución.

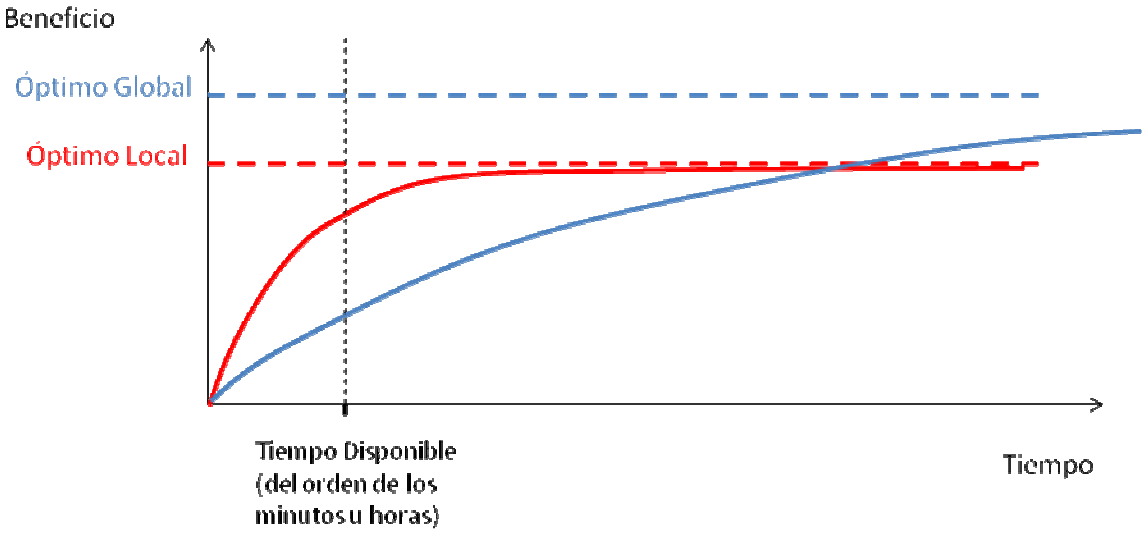

Figura 5.1: Esquema del beneficio hallado en función del tiempo

Como es lógico, el óptimo global es superior al local (en problemas de maximización), con lo cual, si se dispusiera de tiempo infinito, no habría dudas que seguir la curva azul es la mejor opción. Sin embargo, en el mundo real se dispone de cierto tiempo de procesamiento, del orden de algunas horas. Lo que la figura 5.1 intenta esquematizar es que si la optimización se trunca en el "Tiempo Disponible", la curva roja alcanza un mayor beneficio. En este capítulo se verán algunas técnicas para acercar la dinámica de resolución al comportamiento de la curva roja.

Antes de proseguir, es necesario aclarar un concepto fundamental. En este capítulo se propondrán *Variaciones del Problema Completo de Programación de la Producción* (VPCPP) que lo alejan de la definición original. Sin embargo, es absolutamente necesario que el espacio de soluciones factibles de cualquiera de estas variaciones esté dentro del espacio de soluciones factibles del PCPP. En otras palabras, es imposible que una solución factible para una VPCPP no lo sea para el PCPP. De esta forma, se asegura la posibilidad de implementación de cualquier VPCPP o conjunto de éstas dentro del marco del PCPP descripto en el capítulo anterior.

El código de implementación en *CPLEX* de cada VPCPP no se desarrollará por separado. Por el contrario, se desarrollará primero la teoría de cada VPCPP y luego se expondrá un programa en *CPLEX* que nuclea a todas ellas.

5.2 VPCPP 1: Revisión variable y selección de productos

Esta es la única VPCPP cuyo objetivo no es reducir la cantidad de variables binarias a manejar en la resolución del problema. Por el contrario, lo que se busca es reducir la cantidad de operaciones necesarias para el cálculo de costos e ingreso. Esto se realiza de dos maneras, con intervalos de revisión variables y con la selección de productos a tener en cuenta a la hora del cálculo.

5.2.1 Revisión variable

En el cálculo de los costos de almacenamiento y ruptura se puede observar la discretización temporal. Como consecuencia, al aumentar el número de 'utems' del horizonte de estudio la complejidad computacional de calcular estos costos crece de manera acorde. En otras palabras, si uno aproxima la integral mediante trapecios (que es lo que hace la fórmula utilizada), al aumentar el horizonte, aumenta la cantidad de trapecios a calcular.

Lo que propone esta variación es, en lugar de fijar la longitud de la base del trapecio (antes fijada en 1 ‘utem’), fijar la cantidad de trapecios a utilizar. De esta forma, la exactitud en el cálculo depende del tamaño de datos a manejar. Se supondrá que se fija el número en 50 trapecios. Entonces, si el horizonte es de 50 ‘utem’, la base de cada trapecio tendrá longitud 1 ‘utem’. Si, por el contrario, el horizonte es de 500 ‘utem’, cada base medirá 10 ‘utem’ reduciendo a un décimo el requerimiento computacional de calcular los costos comparado con su cálculo con los trapecios de longitud unitaria.

La figura 5.2 esquematiza la implementación de la VPCPP. Notar que al aumentar la longitud temporal, lo que se conserva es la cantidad de trapecios en el cálculo.

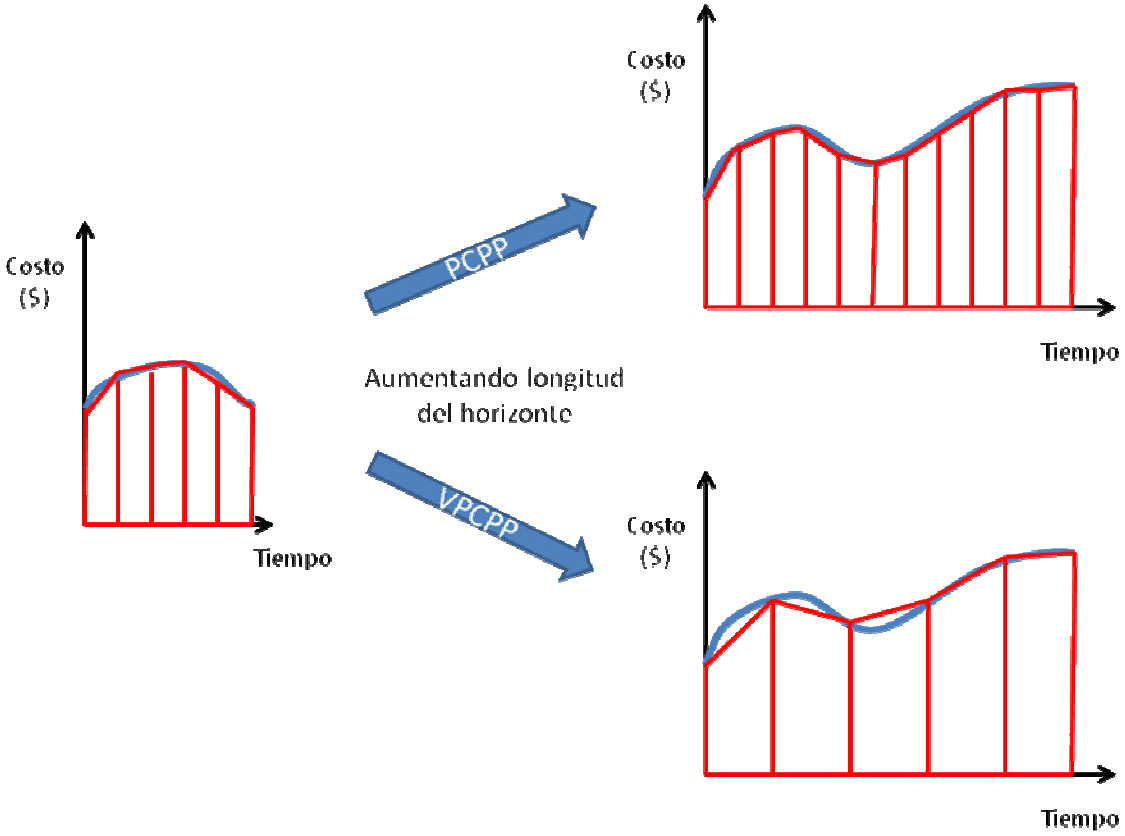

Figura 5.2: Esquematización de la primera VPCPP

5.2.2 Selección de productos

A la hora de calcular el costo de ruptura y los ingresos, no tiene sentido incluir a todos los productos de la fábrica. Esto genera una gran cantidad de operaciones

innecesarias. Cualquier producto que no fue demandado hasta el momento, no puede aportar ni al costo de ruptura ni al ingreso. Por lo tanto, un producto se incluye en el cálculo de éstos para una época dada si tiene demanda no nula durante esa época o durante una época anterior (ya que la demanda se pudo haber acumulado de épocas pasadas). De esta forma, el cálculo se agiliza significativamente ya que todos los productos semi elaborados que no tienen valor en el mercado no se incluyen (suelen ser un número significativo en fábricas de gran tamaño). Además, en el caso en que no todos los productos se demanden durante las primeras épocas, el cálculo se agiliza aún más.

5.3 VPCPP 2: Horas pautadas para Setup

Como se explicó con anterioridad, el tiempo de resolución depende, en gran medida, de la cantidad de variables de decisión enteras (o binarias, en este caso). Más aún, del análisis de éstas surge que la mayoría está asociada al *setup* ya sea a través de MachSetup o de SumaParcialSetup. De esta observación, surge tanto esta VPCPP como la tercera.

Durante el análisis de soluciones de PCPP, es extraño observar una máquina que varíe rápidamente entre los materiales que fabrica. Esto es lógico ya que eso conllevaría un alto costo y una pérdida en capacidad por los tiempos de *setup*. A partir de esto, surge la idea de fijar ciertas 'utem' equidistantes en las cuales las máquinas están posibilitadas para realizar un cambio de producción. En otras palabras, si se denominan éstas como 'utem habilitadas', una máquina durante una 'utem NO habilitada' puede producir únicamente lo que produjo la 'utem' anterior o nada. Por consiguiente, se reduce la cantidad de 'decisiones' que una máquina debe tomar a lo largo del horizonte.

En el modelo, esto repercute aumentando el esparcimiento de la función SumaParcialSetup ya que su valor para cualquier 'utem NO habilitada' pierde sentido (recordar que esta función habilitaba a una máquina a tener la posibilidad de cambiar su producción). Por lo tanto, si se supone una época de 100 'utem' con 20 cambios permitidos (en las 'utem' 5, 10, 15, etc.) la dimensión temporal de SumaParcialSetup se reduce a un quinto de su dimensión original, eliminando gran cantidad de variables binarias. Notar que la dimensión de MachSetup no se modifica ya que, aún cuando el cambio se pueda realizar en 'utem' pautadas, se necesita una cantidad de 'utems' de *setup* anteriores a dicho cambio que dependen de la terna (máquina, producto que se fabricaba, producto que se va a fabricar). Como consecuencia, no se puede eliminar ninguna componente temporal de la función MachSetup.

5.4 VPCPP 3: MachSetup no discriminante

Nuevamente, lo que se busca es reducir la cantidad de variables binarias a manipular. En esta ocasión, se sacrifica el cálculo exacto del costo de *setup* para, a cambio, obtener una gran ganancia en velocidad de procesamiento.

La idea es cambiar el domino de la función MachSetup. En lugar de estar definida sobre el conjunto PosSet, pasa a ser una función binaria de dos variables MachSetup(i,j) que indica si la máquina 'i' está realizando *setup* durante la 'utem j'. Es decir, se pierde la información del par (producto que se fabricó, producto que se va a fabricar).

Si se analiza brevemente cómo impacta este cambio en las restricciones del PCPP, se observa que la restricción RR22 se simplifica ya que el segundo término pasa de ser una sumatoria a ser MachSetup(i,k). La restricción RR24 se mantiene igual, sin perder ningún tipo de información al cambiar el dominio de MachSetup. En la RR29 se pierde información, ya que el costo de *setup* no depende únicamente de la máquina sino de la terna (máquina, producto, producto). Para salvar este inconveniente, se realiza una simplificación y es que la función costo de *setup* (NO el tiempo, sólo el costo) pasa a depender únicamente de la máquina. Para realizar esto, se puede tomar un promedio de los costos para las distintas combinaciones que puede realizar dicha máquina. De esta forma, el costo de *setup* se calcula de manera aproximada, pero se reduce drásticamente el número de variables enteras a manipular.

Es importante comprender que el cálculo aproximado del costo de *setup* no implica que la solución final tendrá el costo mal calculado ya que una vez obtenido el programa de producción óptimo, el cálculo del costo de *setup* es sencillo. Por el contrario, lo que significa es que el costo de *setup* que se utiliza para optimizar (es decir, para comparar dos soluciones) no es calculado de manera exacta.

Se pueden pensar situaciones en las que esta aproximación traiga serios problemas a la hora de implementarla como solución del PCPP. Sin embargo, dichas situaciones distan de la realidad de las fábricas en la generalidad de los casos. Por ejemplo, si para una máquina el costo de un *setup* determinado es órdenes de magnitud superior al promedio para esa misma máquina, entonces el algoritmo no lo distinguiría y penalizaría a cualquier *setup* de dicha máquina por igual. De todas formas, esta situación resulta poco real ya que implicaría, por ejemplo, que pasar de producir A a producir C es mucho más costoso que pasar de A a B y luego de B a C. Aún así, se debe ser consciente de las simplificaciones que se asumen y reconocer su alcance.

A modo de cuantificar la reducción de variables binarias que las VPCPP 2 y 3 generan se considerará la siguiente situación:

"Se tiene una época de 100 'utem' con 20 cambios de producción permitidos, y 3 máquinas que cada una puede producir 4 materiales distintos"

En la tabla 5.1, se detalla la cantidad de variables binarias relacionadas con el *setup* que se deberían manipular en el PCPP y las que se deben manipular al realizar la combinación de VPCPP 2 y 3.

	Cantidad de Variables Binarias	
	PCPP	VPCPP 2 y 3
MachSetup	3*4*3*100 = 3600	3*100 = 300
SumaParcialSetup	3*4*3*100 = 3600	3*4*3*20 = 720
TOTAL	**7200**	**1020**

Tabla 5.1: Cantidad de variables binarias relacionadas con el setup en ambos planteos

Se puede observar que en este ejemplo la cantidad de variables binarias relacionadas con el *setup* se reduce a un séptimo de las originales. Más aún, como se desarrolló con anterioridad, el tiempo de procesamiento depende de manera casi exponencial con la cantidad de variables binarias. De esta forma, para el planteo simultáneo de VPCPP 2 y 3 el tiempo de resolución se reduce drásticamente.

5.5 VPCPP 4: Foco en los Productos Estrella

Con lo descripto hasta aquí, es evidente que lo que se busca es reducir la cantidad de variables enteras a manipular aprovechando el esparcimiento del problema. Al observar varios programas de producción óptimos para distintos datos de entrada de PCPP, es común notar que para una época fija, una máquina dada fabrica únicamente una cantidad limitada de productos distintos, menor a la cantidad posible por diseño. Es decir, si una máquina puede fabricar cinco productos distintos, es común que para cada época, no varíe su producción más allá de, por ejemplo, tres. Esto se debe a que cuando una máquina puede fabricar varios productos, es común que no se pueda satisfacer la demanda de todos ellos. Por lo cual, se concentra en aquellos que dejan mayor margen o que son componentes de productos que dejan un margen elevado. Aún cuando se puede

satisfacer la demanda de todos ellos, es común observar el principio de Pareto donde unos pocos productos representan la mayor parte de los beneficios aportados por dicha máquina. Está claro que los productos a fabricar por cada máquina varían con la época ya que dependen estrechamente del valor de la demanda (que es función de la época).

Por consiguiente, si uno supiese por adelantado cuáles son aquellos productos que cada máquina va a fabricar en cada época, el tiempo hasta encontrar la resolución disminuiría de manera notable ya que los conjuntos PosProd y PosSet tendrían una menor cantidad de elementos. Notar que lo que se debe conocer de antemano es información del tipo cualitativa. En otras palabras, no hace falta conocer la cantidad que se fabricará de cada producto ni su secuencia temporal. Por el contrario, simplemente hace falta conocer si un producto se fabricará o no durante una época dada.

La primera aproximación lógica hacia la determinación del conjunto de productos a fabricar sería a partir del análisis de la información de entrada. Por ejemplo, a partir de los márgenes de ganancia de cada producto y del nivel de demanda en una época dada. El problema de este enfoque es que genera una *'miopía'* en la solución. Si bien no es tan severa como la estudiada en la sección "Múltiples Épocas dentro de un PAPP" porque se optimizaría todo el horizonte de manera global, limitar los productos a fabricar dentro de una época por la información de entrada de la misma hace que la posibilidad de generar stocks para futuras épocas dependa sensiblemente de los datos de entrada.

Para resolver esta complicación, se utilizará un enfoque de resolución en dos módulos: *Inicial* y *Final.* Durante el módulo *Inicial,* se resuelve un PAPP multi-época para determinar los productos más importantes a fabricar por cada máquina en cada época y, luego, se utiliza esa información para alimentar el PCPP y así poder resolverlo más rápido.

Como es evidente, la información de entrada del PAPP no es la misma que la del PCPP, por lo que se deben realizar los cambios necesarios como tiempos de *setup*, costos de *setup, lead-times* e 'inventarios objetivo' nulos.

La estructura de decisión para determinar si un producto se incluye dentro de aquellos "a fabricar" para una época dada está lejos de ser trivial. En este caso, se generó un ranking entre los productos que depende del ingreso que generan en la solución óptima durante la época estudiada y la siguiente (para aumentar la visión de conjunto). Luego, los productos se van incluyendo en el conjunto "a fabricar" a menos que su inclusión genere que una máquina tenga que manejar 'demasiados' productos. Obviamente, el concepto de 'demasiado' se introduce a través de un parámetro, denominado 'MixMachines'. Este parámetro simboliza la mayor

cantidad de productos que cualquier máquina puede manipular en cualquier época.

Es esencial notar que un nuevo producto puede no exceder el límite 'MixMachines' por sí mismo sino por sus componentes. Por ejemplo, si se quisiese incluir el producto A al conjunto "a fabricar" y, hasta el momento, la máquina que lo tiene que producir no tiene ningún producto asignado, pareciera no haber problema. Sin embargo, si el producto A requiere del producto B como componente y la única máquina que lo puede producir ya tiene 'MixMachines' productos asignados, entonces el producto A no se puede incluir en el conjunto "a fabricar".

En la figura 5.3 se puede observar un diagrama de flujo que representa el criterio de decisión para incluir los productos dentro del conjunto "a fabricar".

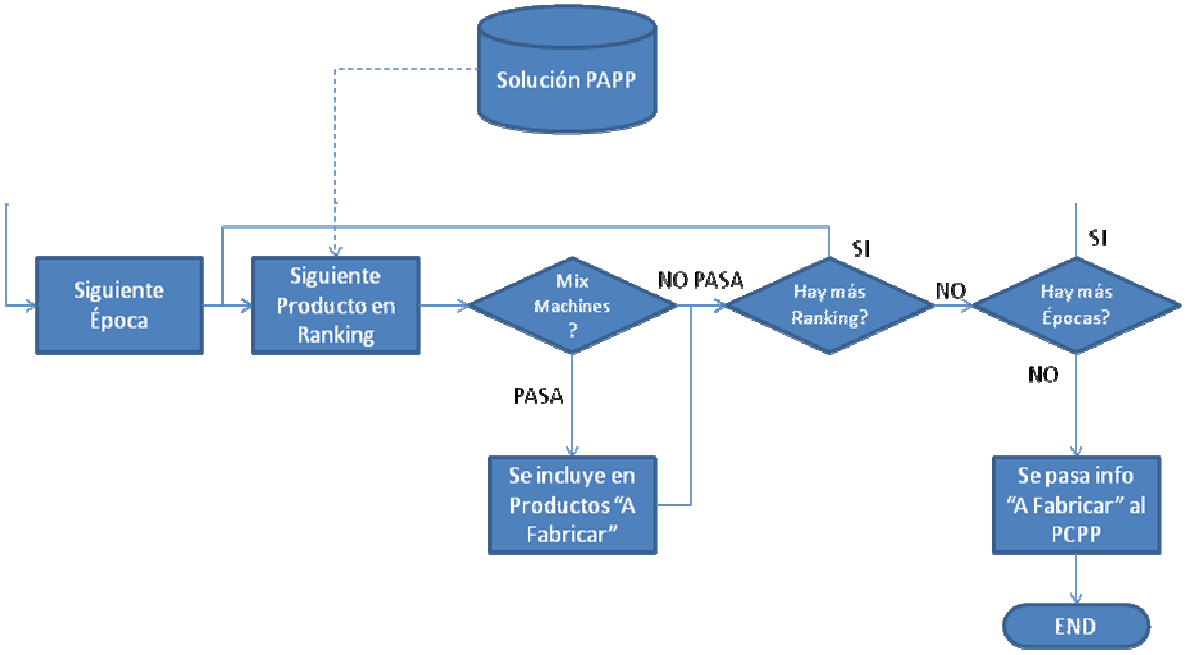

Figura 5.3: Flujograma del criterio de decisión para incluir un producto dentro del conjunto "a fabricar"

La implementación de esta VPCPP es la más complicada ya que presenta numerosos desafíos a la hora de enfrentar su programación. La primera complicación es la corrida secuencial de dos problemas de optimización donde el segundo se alimenta de la solución del primero. Esto requiere de una aplicación externa que 'llame' al modelo de optimización correcto con la información de entrada adecuada en el momento preciso. No es el objetivo del presente trabajo el desarrollo de dicha aplicación sino de los modelos de optimización, para más información sobre el desarrollo de esa parte consultar el proyecto final de carrera de Maximiliano Hense [Hense, 2011].

Otro desafío a la hora de implementar la VPCPP 4 surge de la combinación de la optimización global con la 'miopía' parcial sobre qué se puede producir en cada época. Esto requiere de un 'período de fusión' entre dos épocas consecutivas para asegurar la factibilidad de la solución obtenida. Por ejemplo, si se supone una máquina cuyos productos "a fabricar" para dos épocas consecutivas son conjuntos disjuntos, entonces el vector SumaParcialSetup jamás podría habilitar un cambio de producción entre las dos épocas y la solución carecería de sentido. El 'período de fusión' entre dos épocas combina los conjuntos "a fabricar" de cada una de ellas pudiendo generar una amalgama entre épocas de otra forma incompatibles.

Con el objetivo de asegurar la comprensión de la secuencia de resolución de esta VPCPP se presenta la figura 5.4. Su comprensión es esencial para luego entender la implementación de la combinación de todas las VPCPP.

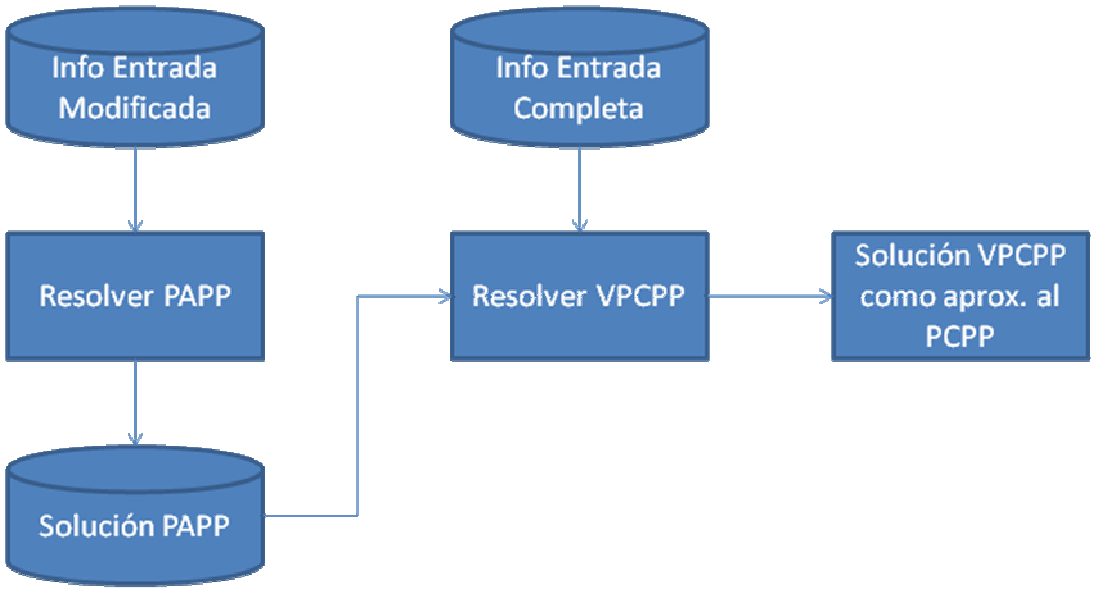

Figura 5.4: Secuencia de resolución de la VPCPP 4

5.6 Implemención de las VPCPP en conjunto

Como es lógico, la ganancia máxima en tiempo de procesamiento se alcanza con la implementación de todas las VPCPP en un mismo programa. Está claro que esto implica realizar todas las aproximaciones discutidas anteriormente de manera simultánea.

Con el objetivo de cuantificar la reducción de variables de decisión binarias totales se tomará el ejemplo utilizado antes con un MixMachines de 3. Es decir:

"Se tiene una época de 100 'utem' con 20 cambios de producción permitidos, y 3 máquinas que cada una puede producir 4 materiales distintos por diseño pero en cada época sólo puede producir un máximo de 3 materiales diferentes."

Sin considerar los 'períodos de fusión' mencionados, cuyo aporte de variables binarias es despreciable, la tabla 5.2 compara la cantidad de variables de decisión binarias en el PCPP y en la combinación de todas las VPCPP.

	Cantidad de Variables Binarias	
	PCPP	Todas VPCPP
MachProd	3*4*100 = 1200	3*3*100 = 900
MachSetup	3*4*3*100 = 3600	3*100 = 300
SumaParcialSetup	3*4*3*100 = 3600	3*3*2*20 = 360
TOTAL	8400	1560

Tabla 5.2: Comparación de la cantidad total de variables de decisión binarias

En este ejemplo, se puede observar una cantidad final inferior a un quinto del número original de variables de decisión binarias. La ganancia en tiempo de resolución es notable, cómo se demostrará a continuación.

Si se corre el mismo set de datos que se utilizó para ejemplificar la solución del PAPP (apéndice B (10.3)) con la combinación de todas las VPCPP, se obtiene un beneficio 0,3% menor (por ser un óptimo local y no global) pero se alcanza en un tiempo de procesamiento que es el 5% del procesamiento original.

Como una segunda prueba del poder de resolución que confiere la combinación de las VPCPP, se resolvió un problema similar al del apéndice B (10.3) pero con una duración más prolongada: a cada una de las cuatro épocas se les asignó una duración de 100 'utem'. Se dejó corriendo el algoritmo de resolución del PCPP y de la combinación de todas las VPCPP durante diez minutos y se graficó el beneficio de la mejor solución hallada en función del tiempo. Además, se graficó el límite superior de la mejor solución entera con líneas punteadas. El resultado se observa en la figura 5.5.

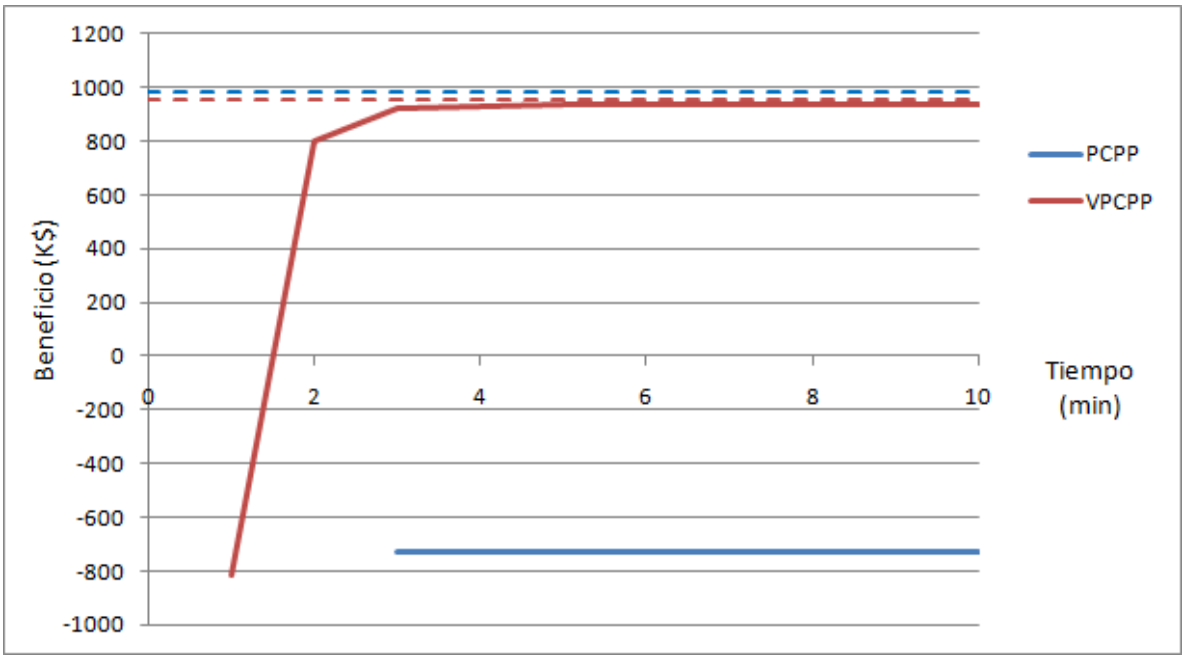

Figura 5.5: Comparación del poder de resolución del PCPP y del VPCPP

En la figura 5.5 se observan varias de las características explicadas hasta el momento. Por un lado, la línea punteada del PCPP es superior a la del VPCPP por tratarse de un máximo global comparado con uno local. Es decir, como la línea punteada representa el límite de la línea sólida, la línea sólida azul superaría a la línea sólida roja para tiempo infinito. Sin embargo, se observa que en el orden de los minutos, el método VPCPP supera ampliamente al PCPP. Por otro lado, se puede observar que las líneas sólidas no comienzan desde el tiempo nulo. Por el contrario, su comienzo se corresponde con el momento en que se halló la primera solución entera factible. Se aprecia que el VPCPP requiere alrededor de un tercio del tiempo que requiere el PCPP en hallar la primera solución. Esto se debe a que al manejar una menor cantidad de variables, el árbol se puede generar con mayor rapidez. Por último, se puede observar una gran diferencia en la velocidad en que cada método recorre el árbol. El PCPP halló su primera solución factible alrededor de los tres minutos y luego no encontró ninguna otra, de allí que en lugar de obtener una curva que tienda a la asíntota, se obtuvo una línea recta. Por otro lado, el VPCPP halló diez soluciones factibles entre los diez minutos (no se llegan a apreciar las diez soluciones en el gráfico por que algunas difieren muy poco de la solución anterior).

Es esencial comprender que la mayoría de las VPCPP son de naturaleza ‘regulable’. Con esto, se intenta transmitir que su poder de ganancia en performance y consecuente alejamiento del óptimo global se puede regular con parámetros. Ejemplos de esto son el parámetro MixMachines y la cantidad de

setups permitidos por época. De esta forma, el usuario puede modificar los parámetros para que se ajusten mejor a su fábrica, su tiempo y su capacidad de procesamiento. En consecuencia, la herramienta desarrollada se puede configurar para ser tanto útil corriendo diez minutos en una computadora personal o un día entero en una super computadora, simplemente modificando unos pocos parámetros. Esta flexibilidad es un aporte clave para la universalidad de la aplicación de la herramienta, uno de los principales objetivos del trabajo.

En los apéndices A (9.5 y 9.6) se encuentran los códigos en *CPLEX* para la aplicación de lo desarrollado en este capítulo. Las restricciones no se numeran como en los códigos anteriores. Se incluyen comentarios para facilitar la comprensión del código.

En el apéndice A (9.5) se explicita el código del módulo *Inicial,* es decir, aquél que resuelve el PAPP para obtener la información cualitativa de qué productos fabricar en cada época para cada máquina. Se incluye, además, el módulo de post procesamiento que extrae esta información en forma de vector.

En el apéndice A (9.6) se encuentra el código del módulo *Final,* es decir, aquél que toma la información completa de entrada y aquella encontrada por el módulo *Inicial* y resuelve el problema original. Las variables generadas para incluir las distintas VPCPP están marcadas con el correspondiente número.

6 LIMITACIONES Y FUTUROS PASOS

Resulta evidente que la herramienta desarrollada posee ciertas limitaciones tanto en su aplicabilidad como en su habilidad de representar la realidad. Algunas de estas limitaciones se mencionaron a lo largo del trabajo mientras que otras permanecieron implícitas. En la siguiente sección se explicarán las mayores limitaciones que la herramienta muestra así como un esbozo de los pasos a seguir para superarlas.

6.1 Limitaciones

Siendo su objetivo último la aplicación para la resolución de problemas integrales dentro de la empresa, su principal limitación es la falta de integración con el resto del proceso logístico. Es decir, el hecho de suponer abastecimiento infinito y logística de distribución perfecta dista significativamente de la realidad. Por ejemplo, un programa que optimiza el beneficio para la fábrica desde el punto de vista considerado a lo largo de todo este trabajo puede no ser viable por factores que exceden el ámbito productivo. Por ejemplo, puede ocurrir que la materia prima no se consiga en las cantidades necesarias o que, de conseguirlas, haya que pagar un precio extra que modifique el programa óptimo. Por otro lado, en el otro extremo de la cadena, podría ocurrir que la empresa no cuente con capacidad de distribución suficiente para entregar los productos del programa óptimo teniendo que tercerizar parte de la distribución incurriendo en gastos no considerados. Se pueden nombrar otros problemas, pero la raíz de todos ellos sería la falta de integración de la solución con el resto de la cadena logística.

Como dato de entrada, se requiere la demanda futura de los distintos productos. Es evidente que esta es una información inexacta y difícil de predecir. Sin embargo, existen herramientas informáticas que facilitarían el trabajo e incrementarían la validez de los datos. Aún así, la herramienta desarrollada no incluye ninguna ayuda para facilitar al usuario el ingreso de este tipo de información.

Otra limitación, resulta ser el determinismo de los datos de entrada. Es decir, los datos de entrada se consideran información exacta y no se permite variación sobre estos. Si bien la limitación más importante se da con el nivel de demanda, muchos otros datos de entrada sufren de esta limitación. Por ejemplo, el ritmo de producción. Cuando se introduce como dato de entrada que la máquina ‘X’ produce el material ‘A’ a un ritmo de r [ust/utem], implica que este ritmo se respetará de manera exacta durante todo el horizonte de estudio. Esta claro que

esto dista de la realidad, ya que el ritmo sufre pequeñas variaciones dependiendo de varios factores como el medio ambiente, la experiencia de los operarios y la calidad de la materia prima.

Si bien el concepto de 'máquina' se presentó como más amplio que el término literal, es decir, que puede incluir al conjunto 'máquina-operario', la herramienta no considera la disponibilidad de mano de obra por separado. Es decir, no concibe la posibilidad de que una máquina no pueda ponerse en marcha por falta de un operario que realice la acción. Se podría representar parcialmente este tipo de situaciones afectando la capacidad horaria de las máquinas, pero sería interesante incluir una solución de fondo que ayude al dimensionamiento de la mano de obra.

Como consecuencia de aceptar los determinismos, la herramienta sufre de una incapacidad de manejar contingencias menores. Por ejemplo, si se posee un programa óptimo donde a una máquina se le había asignado una capacidad horaria un poco superior de la que terminó disponiendo realmente (por que el mantenimiento llevó más tiempo del esperado, por ejemplo). En este caso, la herramienta no aporta ningún indicio sobre cuál es el mejor camino a tomar por el programador para modificar el programa. Se puede correr toda la optimización desde el principio con la nueva capacidad, pero esto resultaría poco eficiente en un ambiente productivo dinámico.

Por otro lado, se puede considerar la semejanza del modelo representado con un proceso productivo real. En primer lugar, la estructura del proceso productivo no es perfectamente flexible. Si bien se permite variar el número de etapas productivas y el de máquinas en cada etapa, se impone que el input de una etapa provenga de la etapa exactamente anterior, quitando flexibilidad al diseño del proceso productivo. Más aún, la herramienta está pensada para programas de producción por lotes flexibles con almacenamientos intermedios. Por consiguiente, su aplicación a procesos productivos continuos no es inmediata. Sin embargo, la resolución de problemas de esta característica pareciera ser, a priori, más sencilla por poseer un menor número de decisiones a tomar. Además, existe una lista inagotable de detalles que no se tienen en cuenta y que alejan al modelo de la realidad. Por nombrar algunos, no se consideran desperdicios o productos secundarios de los procesos productivos, no se incluyen controles de calidad entre procesos, no se ofrecen descuentos por cantidad en los precios, los costos de producción varían de forma proporcional con la cantidad producida sin tener en cuenta la estructura fija de costos, no se consideran mejoras por curva de aprendizaje de los operarios, los espacios de almacenamiento intermedio se consideran ilimitados, los tamaños de los lotes son perfectamente flexibles y los lotes no poseen tiempo de caducidad en inventario.

Por último, la alta complejidad del enfoque adoptado para asegurar la factibilidad del programa de producción limita el horizonte de estudio posible. Es decir, resolver el problema completo al nivel de resolución del programa de producción hace dificultosa la optimización de largos horizontes aún con las estrategias diseñadas para mejorar la performance explicadas en la sección precedente. Dependiendo de las necesidades del usuario, esta limitación podría representar un problema para su aplicación.

6.2 Futuros pasos

La idea de esta sección es presentar posibles alternativas de investigación que ayuden a superar las limitaciones mencionadas anteriormente.

Probablemente la línea de investigación más interesante y desafiante es la integración de la herramienta desarrollada en un sistema integral de soluciones empresariales tipo ERP o ASP. Es decir, además de la evidente integración con los sistemas de abastecimiento y logística de distribución, alcanzar una integración total con el resto de los sistemas. Por ejemplo, que el dimensionamiento de la mano de obra (otro de los pasos a realizar) alimente el sistema de recursos humanos, que el programa óptimo de abastecimiento nutra al sistema de compras y así para el resto de los sistemas. Está claro que este es un objetivo lejano para la presente herramienta, pero debe ser el norte hacia el cual los desarrollos futuros deben apuntar. De esta forma, se promueve la aplicabilidad y, por consiguiente, la adopción en el ámbito empresarial.

Para facilitar al usuario el input de las futuras demandas, a la herramienta se le puede añadir un módulo de ‘pronóstico de la demanda’. Este módulo puede utilizar análisis de históricos para descubrir tendencias y estacionalidades y así ayudar al usuario a la hora de introducir los valores de la demanda. De más está decir que el usuario debe ser capaz de modificar la demanda pronosticada valiéndose de información que exceda el ámbito estadístico.

Escapar del determinismo en los datos de entrada no es algo sencillo. Una posible línea de investigación es el método de Monte Carlo. En este método, los datos de entrada se introducen como variables aleatorias con una distribución de probabilidad y, en cada corrida, la variable toma un valor dado y se resuelve el problema. Luego de un alto número de corridas, se obtiene un histograma para la función resultado (el beneficio económico, por ejemplo) que considera la posible variación en los datos de entrada. Si bien esta solución es muy elegante, a priori la complejidad de cálculo pareciera exceder las capacidades de cómputo actuales. Sin embargo, puede ser una línea de investigación interesante y fructífera.

Incluir el dimensionamiento de la mano de obra no parece ser, a priori, una tarea muy compleja y aumentaría la utilidad de la herramienta significativamente. Sería necesario incluir como datos de entrada el ritmo de trabajo promedio de los distintos tipos de operarios, la longitud de los turnos, los salarios y demás. Se deben explicitar las restricciones necesarias sobre capacidad de la mano de obra e incluir los costos en el cálculo del beneficio.

Para aumentar la respuesta de la herramienta ante las pequeñas contingencias que pueden surgir en el día a día, resulta interesante estudiar las variaciones marginales sobre programas ya existentes. Es decir, en lugar de resolver todo el problema desde cero, brindarle a la herramienta la solución antes de la contingencia como dato de entrada y el cambio en los datos generado por la contingencia (por ejemplo, menor capacidad de máquina). De esta forma, la herramienta puede variar rápidamente la solución anterior hacia el nuevo óptimo factible.

El acercamiento del modelo a la realidad conlleva dos principales problemas. Por un lado, las nuevas características (desperdicio, controles de calidad, límites espaciales, lotes predeterminados) se deben incluir en el modelo mediante restricciones. El diseño de dichas restricciones en armonía con todo el resto del modelo está lejos de ser trivial. Por otro lado, las nuevas características aumentarían la complejidad del cálculo y, por consiguiente, el tiempo de procesamiento. En este punto no queda del todo claro si un acercamiento del modelo a la realidad sería una ayuda por brindar una solución más completa o un problema por requerir una solución demasiado difícil de alcanzar en un tiempo razonable. Por consiguiente, resulta lógico tomar como estrategia hacia el futuro la inclusión selectiva de estas características dependiendo de la industria. Por ejemplo, en una industria alimenticia, el tiempo de vencimiento de los lotes puede ser una característica fundamental a la hora de optimizar el programa de producción mientras que para una metalúrgica probablemente no. Por lo tanto, el agregado de dicha característica resultaría de gran valor para el primero pero una pérdida de tiempo para el segundo. De allí que se toma como estrategia la inclusión futura selectiva de distintas características para el mejor amoldamiento a industrias en particular.

Otro punto desafiante es modificar ligeramente la herramienta para poder resolver programas de producción en procesos continuos. El planteo debería ser distinto ya que no se toman decisiones en todas las etapas productivas al no haber almacenes intermedios. Es decir, la actividad de la primera etapa determina todas las subsiguientes. Esto indica una resolución posiblemente más sencilla, pero es un campo que definitivamente vale la pena explorar.

Por último, si se quiere que la herramienta funcione correctamente para largos horizontes de estudio, es necesario cambiar el enfoque. En el comienzo del trabajo se explicó que se estudia todo el problema desde la alta resolución del programa de producción para asegurar su factibilidad y se logra su satisfactoria aplicación gracias a las estrategias que reducen la complejidad en el cálculo. Sin embargo, si se buscan resolver problemas de mayor envergadura, es necesario un paso intermedio similar al plan de producción (podría ser un programa de menor resolución) y que éste alimente la optimización del programa. Esto volvería al viejo problema de no asegurar la factibilidad del programa de producción. Por eso, sería interesante trabajar sobre cambios marginales en la solución similares a los mencionados para el manejo de contingencias. De esta forma, si se llega a una solución no factible con un plan dado, se podrían ir relajando las restricciones de producción hasta hallar una factible. Si bien este estudio requiere un conocimiento de los algoritmos de resolución superior a los indicados en este trabajo, su explotación puede ser de gran ayuda para resolver problemas de grandes dimensiones.

A continuación, en la tabla 6.1 se presenta un cuadro que resume las limitaciones mencionadas y las correspondientes líneas de investigación para superarlas. Recordar que esto no implica que no existan otras limitaciones u otras formas de superar las mencionadas, simplemente se intentan mostrar algunas posibilidades de mejora y generar disparadores para futuras investigaciones.

Limitaciones	Futuros Pasos
Optimización aislada de las actividades dentro de la planta	Integración tanto con la cadena de abastecimiento, la logística de distribución y otros sistemas informáticos dentro de la empresa
Incerteza de la demanda futura como dato de entrada	Módulo de 'pronóstico de demanda' analizando datos históricos
Determinismo en la información de entrada	Método de Monte Carlo
Factor humano no considerado en las restricciones del modelo	Incluir el dimensionamiento de la mano de obra como parte de la solución
No aporta información para el manejo de contingencias menores	Variaciones marginales de la solución para pequeños cambios en los datos de entrada
El modelo matemático no se acerca perfectamente a la realidad	Inclusión selectiva de un mayor número de características en las restricciones. (Cuidado con el crecimiento en complejidad)
No se consideran procesos productivos continuos	Adaptar la herramienta para que pueda resolver programas de producción de procesos continuos
No alcanza a resolver problemas de gran envergadura en un tiempo razonable	Comenzar por un nivel de agregación mayor y luego asegurar la factibilidad del programa mediante una relajación selectiva de las restricciones

Tabla 6.1: Resumen de las limitaciones y las posibles líneas de futura investigación

7 CONCLUSIÓN

La optimización del plan y programa de producción para un proceso productivo universal resulta demasiado compleja de abordar y su definición es ambiguamente amplia. Por consiguiente, el presente trabajo se focalizó en desarrollar una herramienta para hallar un programa de producción factible y cercano al óptimo en un tiempo razonable aplicando programación lineal entera mixta para resolver un problema con datos de entrada flexibles pero formato fijo.

El desarrollo tuvo como principal objetivo promover la universalidad y la aplicabilidad de la herramienta en el mundo cotidiano de la ingeniería industrial. Con el objetivo de alcanzar la primera propiedad, se generó una herramienta que acepta la introducción de las características del proceso productivo así como las especificaciones de las máquinas y los productos por parte del usuario. De esta forma, aún sin un alcance absoluto, la herramienta resulta útil para distintos tipos de industrias y de configuraciones productivas. Para garantizar su aplicabilidad, se trabajó sobre la cercanía del modelo matemático con la realidad y el tiempo necesario de resolución. En cuanto al modelo matemático, considera aspectos productivos de gran relevancia ignorados por otros trabajos similares como los *lead-times* y los tiempos y costos de *setup.* El tiempo de resolución resulta un aspecto clave en cualquier ambiente productivo dinámico por lo que se trabajó sobre éste desarrollando estrategias *ad-hoc*. Estas estrategias logran maximizar el beneficio en el delicado equilibrio entre pérdida de exactitud y ganancia de performance.

La combinación del modelo matemático con la serie de estrategias desarrolladas para aumentar su performance se ha aplicado computacionalmente resolviendo de manera exitosa una serie de ejemplos distintos. Más aún, el valor de la información de salida obtenida ha superado el simple programa o plan de producción para brindar información a diferentes áreas de la empresa. Estos datos han demostrado ser de gran utilidad para distintos niveles gerenciales, desde el programa detallado de producción que guía las actividades de la planta a nivel operativo hasta la facturación consolidada para guiar estrategias de marketing entre la cartera de productos.

Aún así, resulta evidente que el estado actual de desarrollo no permite su adopción directa por la industria. Para alcanzar dicho objetivo, se tendrían que superar una serie de limitaciones mencionadas a lo largo del trabajo, siendo la principal de ellas la falta de integración con otros sistemas de información como el de la cadena logística. Además, resulta necesario aumentar el valor de la herramienta para distintos tipos de industrias mediante la inclusión selectiva de

características distintivas. De este modo, se poseería una batería de opciones a agregar (lotes de tamaño fijo, manejo de desperdicios, lotes con vencimiento, etc.) dependiendo de la industria en que se aplica. La inclusión simultánea de todas ellas no sería recomendable por aumentar notablemente la complejidad de la solución. Más allá de la imposibilidad de la inmediata adopción, el trabajo explora algunos aspectos innovadores dentro de la resolución del problema general de planeamiento y programación de la producción, realizando su aporte hacia la solución definitiva de uno de los problemas más complejos y vigentes de la investigación operativa dentro de la ingeniería industrial.

8 BIBLIOGRAFÍA

Anily S., Tzur M., Wolsey L. 2008. Multi-item lot-sizing with joint set-up costs. Springer.

Barták R. 2000. Dynamic Constraint Models for Planning and Scheduling Problems. LNAI Series.

Brusoni V., Console L., Lamma E., Mello P., Milano M., Terenziani P. 1996. Resourcebased vs. Task-based Approaches for Scheduling Problems. Proceedings of the 9th ISMIS96. LNCS Series. Springer Verlag.

Cai Y., Kutanoglu E., Hasenbein J. 2011. Production Planning and Scheduling: Interaction and Coordination. Planning production and inventories in the extended enterprise: a state-of-the-art handbook, volume 2. Chapter 2. Springer.

Dai J., Weiss G. 2002 A fluid heuristic for minimizing makespan in job shops. Oper Res 50(4), p 692–707

Florian M., Lenstra J.K., Rinnooy Kan A.H.G. 1980. Deterministic Production Planning: Algorithms and Complexity. Management Science, 26, 7, p 669-679.

Forrester J.W. 1958. Industrial dynamic. M.I.T. Press. Cambridge.

Hackman S., Leachman R. 1989. A general framework for modeling production. Manag Sci 35:478–495.

Hense, M. 2011. Creación de un software para planificación de producción. Proyecto Final de Ingeniería Industrial.

Hicks D.A. 1997. The manager´s guide to supply chain and logistics problem-solving tools and techniques. IIIE Solutions. Ed. 10, p 24-29

Hung Y.F., Leachman R.C. 1996. A production planning methodology for semiconductor manufacturing based on iterative simulation and linear programming calculations. IEEE Trans Semicond Manuf 9(2), p 257–269.

IBM ILOG OPL V6.3. 2009. IBM ILOG OPL Language User's Manual.

ILOG OPL Development Studio. 2007. Optimization with ILOG OPL Development Studio: Student Workbook.

Lee Y., Kim T. 2002. Manufacturing cycle time reduction using balance control in the semiconductor fabrication line. Prod Plann Contr 13, p 529–540

Luenberger D., Ye Y. 2008. Linear and Non Linear Programming. Springer.

Mikosch T., Resnick S., Robinson S. 2006. Production Planning by Mixed Integer Programming. Springer.

Monk E., Wagner B. 2006. Concepts in Enterprise Resource Planning. 2nd Edition. Editor Mac Mendelsohn. Canada.

Pool D., Mackworth A., Goebel R. 1998. Computational Intelligence – A Logical Approach. Oxford University Press. Oxford.

Real Academia Española. 2011. Diccionario de la Lengua Española 2.0.

Sheridan J.H. 1995. *Which path to follow*. Industry Week. Vol 244. N°13, p 41-46

Sipper D., Bulfin R. L. 1998. Planeación y Control de la Producción. Mc Graw Hill.

Van Eck M. 2003. Advanced Planning and Scheduling. BWI paper

Winston W. 2005. Investigación de Operaciones. Thomson.

Wolsey, L. 2006. Production Planning by Mixed Integer Programming. Springer.

9 APÉNDICE A: CÓDIGOS DE APLICACIÓN EN IBM ILOG CPLEX

9.1 Ejemplo del impacto del esparcimiento

9.1.1 Resolución sin explotar el esparcimiento

```
int d = 5000;
range machines = 1..d;
range products = 1..d;
int produce[machines][products];
execute CARGAR_PRODUCE{
	for (var i in machines){
		for (var j in products){
			if (i-j==0){
				produce[i][j]=1;
			}
		}
	}
}
execute IMPRIMIR{
writeln ("Combinaciones permitidas");
	for (var i in machines){
		for (var j in products){
			if (produce[i][j]==1){
				writeln("Maquina ", i, " Producto ", j);
			}
		}
	}
}
```

9.1.2 Resolución explotando el esparcimiento

```
int d = 5000;
tuple produccion{
	int machine;
	int product;
}
range machines = 1..d;
range products = 1..d;
{produccion} conjuntoreducido = {<i,j>| i in machines, j in products: i-
j==0};
execute IMPRIMIR{
writeln ("Combinaciones permitidas");
	for (var a in conjuntoreducido){
				writeln("Maquina ", a.machine, " Producto ",
a.product);
	}
}
```

9.2 Resolución del PAPP

```
/* ACLARACIÓN---------------------------------------------------------
Las tuples son estructuras de datos que permiten explotar el
esparcimiento del problema. El presente está lejos de ser un manual de
IBM ILOG CPLEX por lo que aspectos puntuales del lenguaje de programación
(como las tuples) no serán explicados en detalle.
-------------------------------------------------------------------------*/
// Definición de Tuples ----------------------------------------------
tuple TPpsBom {
  string material;
  string component;
  float factor;
};

tuple TPpsDemand {
  string material;
  int duration;
  float demand;
};

tuple TPpsMachines {
  int stage;
  key string machine;
  float nethours;
};

tuple TPpsMaterialprod {
  string machine;
  string material;
  float prodrate;
  float prodcost;
};

tuple TPpsMaterials {
  key string material;
  float price;
  float target;
  float holdingcost;
  float backordercost;
};

tuple TPpsParameters {
  key string parameterid;
  float parametervalue;
};

// Fin definición de tuples ------------------------------------------
// Cargar tuples de archivo .dat -------------------------------------
{TPpsBom} PPS_BOM = ...;
{TPpsDemand} PPS_DEMAND = ...;
{TPpsMachines} PPS_MACHINES = ...;
{TPpsMaterialprod} PPS_MATERIALPROD = ...;
{TPpsMaterials} PPS_MATERIALS = ...;
{TPpsParameters} PPS_PARAMETERS = ...;
// Comienzo creación de variables ------------------------------------
```

```
int var_horizon;
execute PARAMETERS{
      var_horizon = PPS_PARAMETERS.find("Horizon").parametervalue;
}
range horizon=1..var_horizon;
float machinecapacity[PPS_MACHINES];
execute MACHINECAPACITY{
      for (var a in PPS_MACHINES){
            machinecapacity[PPS_MACHINES.find(a.machine)] =a.nethours;
      }

}
// Definir el conjunto "PosProd" para explotar el esparcimiento ---------
tuple prueba{
      string machine;
      string material1;
      int time;
}

setof(prueba) PosProd={<i,j,k> | <i,j,l,m> in PPS_MATERIALPROD, k in
horizon};
// Comenzar la creación de variables de decisión-------------------------
/* Notar que el signo '+' indica que la variable debe ser no negativa.
Por lo tanto, se introducen restricciones en la definición de la variable
lo que aumenta el rendimiento del algoritmo de resolución.*/
dvar float mach_prod[PosProd] in 0..1; // R6
dvar float+ stock [PPS_MATERIALS][horizon]; // R1
dvar float+ sales [PPS_MATERIALS][horizon]; // R2
dvar float+ backorder [PPS_MATERIALS][horizon]; // R3
dvar float+ produ [PPS_MATERIALS][horizon]; // R4
dvar float+ use[PPS_MATERIALS][horizon]; // R5
dvar float+ hold_cost; // R7
dvar float+ back_cost; // R9
dvar float+ prod_cost; // R8
dvar float+ tot_revenue; // R10
dexpr float total_benefit = tot_revenue - hold_cost - back_cost-
prod_cost;
// Finalizar definición de variables de decisión ------------------------
// Función Objetivo -----------------------------------------------------
maximize total_benefit;
// Comienzo Restricciones -----------------------------------------------
subject to{

      forall (i in PPS_MATERIALS){
            stock[i][1]==produ[i][1]-sales[i][1]-use[i][1]; // R11
            backorder[i][1] + sales[i][1] == sum(h in
PPS_DEMAND:h.material==i.material)h.demand; //R13
            use[i][1]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][1]*b.factor; // R15
            produ[i][1]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, 1>])*n)); // R16
      forall (k in horizon:k>1){
            stock[i][k]==stock[i][k-1]+produ[i][k]-sales[i][k]-use[i][k];
// R12
                  backorder[i][k] + sales[i][k] == backorder[i][k-
1]+sum(h in PPS_DEMAND:h.material==i.material)h.demand;    // R14
```

```
                  use[i][k]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][k]*b.factor;// R15
            produ[i][k]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, k>])*n));// R16
            }

      }

// R17
      forall (k in horizon){
            forall(i in PPS_MACHINES){
                  sum(<i.machine, j, k> in
PosProd)mach_prod[<i.machine,j,k>]<=1;
            }
      }
// R18
      forall (i in PPS_MACHINES){
            sum(<i.machine, j, k> in
PosProd)mach_prod[<i.machine,j,k>]<=machinecapacity[i];
      }
// Cálculo de costos e ingreso ----------------------------------------
// R19
      hold_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)stock[i][k]*i.holdingcost+sum(i in PPS_MATERIALS)
stock[i][var_horizon]*0.5*i.holdingcost;
// R20
      back_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)backorder[i][k]*i.backordercost+sum(i in PPS_MATERIALS)
backorder[i][var_horizon]*0.5*i.backordercost;
// R21
      prod_cost==sum(p in PPS_MATERIALPROD, <p.machine, p.material, k> in
PosProd) (p.prodcost*p.prodrate*(mach_prod[<p.machine, p.material, k>]));
// R22
      tot_revenue==sum(i in PPS_MATERIALS, k in horizon)
sales[i][k]*i.price;
}
// Fin Restricciones --------------------------------------------------
// FIN DE PROGRAMA ----------------------------------------------------
```

9.3 Comparación entre unidades continuas y discretas

9.3.1 Problema para comparar el valor del funcional en ambos casos

```
range Prod = 1..10;
range Mat = 1..10;
int compo[Prod][Mat] =
[[1,1,0,0,0,1,3,2,1,0],[0,2,0,0,1,2,0,1,0,0],[0,0,0,1,1,0,5,1,0,1],[0,0,3
,1,1,0,0,0,2,0],[0,1,1,0,1,1,1,1,1,0],[0,1,1,0,2,0,1,3,0,0],[4,0,0,2,1,0,
1,0,1,0],[0,2,1,1,0,0,0,0,6,0],[0,7,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0,3
]]; // Ejemplo inventado con el que se corrió el programa
int margen[Prod] = [47, 31, 33, 52, 47,49,71,37,22,13];
int cantidad[Mat] = [1002,801,1302,1233, 921,1253,1367,587,911,1217];
dvar int+ producir[Prod]; // decidir si es entero o continuo
maximize sum(p in Prod)producir[p]*margen[p];
subject to{
      forall(m in Mat){
            sum(p in Prod)producir[p]*compo[p][m]<=cantidad[m];
      }
}
```

9.3.2 Problema para comparar el tiempo de resolución en ambos casos

```
int d = 10;
range Prod = 1..d;
range Mat = 1..d;
int compo[Prod][Mat];
execute COMPO{
for (var a in Prod){
      for (var b in Mat){
            compo[a][b]=Opl.rand(4);
      }
}
}
int margen[Prod];
execute MARGEN{
for (var a in Prod){
      margen[a]=10+Opl.rand(40);
}
}
int cantidad[Mat];
execute CANTIDAD{
for (var b in Mat){
      cantidad[b]=500+Opl.rand(800);
}
}
dvar int+ producir[Prod];
maximize sum(p in Prod)producir[p]*margen[p];
subject to{
      forall(m in Mat){
            sum(p in Prod)producir[p]*compo[p][m]<=cantidad[m];
      }
}
```

9.4 Resolución del PCPP

```
// Definición de Tuples -------------------------------------------
tuple TPpsBom {
  string material;
  string component;
  float factor;
};

tuple TPpsDemand {
  string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsDemand2 {
  key string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsLeadtimes {
  int stage1;
  int stage2;
  int leadtime;
};

tuple TPpsMachines {
  int stage;
  string machine;
  int epoch;
  float nethours;
};

tuple TPpsMachines2 {
  int stage;
  key string machine;
  int epoch;
  float nethours;
};

tuple TPpsMaterialprod {
  string machine;
  string material;
  float prodrate;
  float prodcost;
};

tuple TPpsMaterials {
  key string material;
  float price;
  float target;
  float holdingcost;
  float backordercost;
```

```
};

tuple TPpsParameters {
  key string parameterid;
  float parametervalue;
};

tuple TPpsSetup {
  string machine;
  string materialFrom;
  string materialTo;
  int setuptime;
  float setupcost;
};
// Fin definición de tuples ------------------------------------------
// Cargar tuples de archivo .dat -------------------------------------
{TPpsBom} PPS_BOM = ...;
{TPpsDemand} PPS_DEMAND = ...;
{TPpsLeadtimes} PPS_LEADTIMES = ...;
{TPpsMachines} PPS_MACHINES = ...;
{TPpsMaterialprod} PPS_MATERIALPROD = ...;
{TPpsMaterials} PPS_MATERIALS = ...;
{TPpsParameters} PPS_PARAMETERS = ...;
{TPpsSetup} PPS_SETUP = ...;
// Comienzo creación de variables ------------------------------------
//Crear una variable con el número de épocas "num_epochs" y un rango
//desde 1 al número de épocas: "epochs" -----------------------------
int num_epochs;
execute NUM_EPOCHS{
 for(var s in PPS_DEMAND){
      if(num_epochs<s.epoch){
            num_epochs=s.epoch;
      }
 }
}
range epochs=1..num_epochs;
//Dividir DEMANDA y CAPACIDAD en conjuntos basados en el número de época
{TPpsDemand2} ppsDemand2[i in epochs] = {<a.material, a.epoch,
a.duration, a.demand>|a in PPS_DEMAND:a.epoch==i};
{TPpsMachines2} ppsMachines2[i in epochs] = {<a.stage, a.machine,
a.epoch,a.nethours>|a in PPS_MACHINES:a.epoch==i};
//Crear un vector con la duración de las épocas ----------------------
int dur_epochs[epochs];
execute DUR_EPOCHS {
      for (var e in epochs){
            for (var s in ppsDemand2[e]){
                  if (e == s.epoch){
                        dur_epochs[e]=s.duration;
                        break;
                  }
            }
      }
}
//Cargar el horizonte, el número de etapas y el FDEFICIT ---------------
int var_horizon;
int num_stages;
float inv_deficit;
```

```
execute PARAMETERS{
	var_horizon = PPS_PARAMETERS.find("Horizon").parametervalue;
	num_stages = PPS_PARAMETERS.find("Stages").parametervalue;
	inv_deficit = PPS_PARAMETERS.find("InvDeficitCost").parametervalue;
}
range horizon=1..var_horizon;
range stages = 1..num_stages;
//Cargar leadtimes en un vector ----------------------------------------
int leadtimes[stages];
execute LEADTIMES{
	var max;
	for (var s in stages){
		max = 0;
		for(var l in PPS_LEADTIMES){
			if(l.stage1==s && l.leadtime>max){
				max=l.leadtime;
			}
		}
		leadtimes[s]=max;
	}
	leadtimes[num_stages]=0;
}
//Determinar el máximo leadtime (más adelante se usa para salvar un error
//de índice negativo) ------------------------------------------------
int max_lead;
execute MAX_LEAD{
	for (var s in stages){
		if (max_lead<leadtimes[s]){
			max_lead=leadtimes[s];
		}
	}
}
//Determinar la etapa de producción de cada material -------------------
int mat_stage[PPS_MATERIALS];
execute MAT_STAGE{
	for (var m in PPS_MATERIALS){
	mat_stage[m]=1;
		for (var p in PPS_MATERIALPROD){
			if (m.material == p.material){

	mat_stage[m]=ppsMachines2[1].find(p.machine).stage;
				}
			}
		}
	}
// Generar un vector con las capacidades de cada máquina en cada época --
float machinecapacity[ppsMachines2[1]][epochs];
execute MACHINECAPACITY{
for (var e in epochs){
	for (var a in ppsMachines2[e]){
	machinecapacity[ppsMachines2[1].find(a.machine)][e]=a.nethours;
	}
}
}
//---------------------------------------------------------------------
// Horizon2 es un rango que tiene algunos indices negativos
// Se utiliza para el vector 'Produ' para evitar problemas de índice
```

```
// negativo en el vector 'Disponible' ya que este último mira al vector
// 'Produ' lead time utems en el pasado.
range horizon2 = -max_lead..var_horizon;
//"start" contiene las 'utem' de comienzo de cada época ----------------
int start[1..num_epochs+1];
execute START{
      start[num_epochs+1]=var_horizon+1;
      start[1]=1;
      for (var e in epochs){
            if (e>1){
            start[e] = start[e-1] + dur_epochs[e-1];
            }
      }
}
// Se definen PosProd y PosSet para explotar el esparcimiento ----------
tuple seteo{
      string machine;
      string materialFrom;
      string materialTo;
      int time;
}
tuple prueba{
      string machine;
      string material1;
      int time;
}

setof(prueba) PosProd = {<i,j,k> | <i,j,l,m> in PPS_MATERIALPROD, k in
horizon};
setof(seteo) PosSet = {<i,j,l,k> | <i,j,l,m,n> in PPS_SETUP, k in
horizon};
/* Comenzar la creación de variables de decisión. Notar que el signo '+'
indica que la variable debe ser no negativa. Por lo tanto, se introducen
restricciones en la definición de la variable lo que aumenta el
rendimiento del algoritmo de resolución.*/
dvar boolean mach_prod[PosProd];
dvar boolean mach_setup[PosSet];
dvar float+ stock [PPS_MATERIALS][horizon]; //RR1
dvar float+ sales [PPS_MATERIALS][horizon]; //RR2
dvar float+ backorder [PPS_MATERIALS][horizon]; //RR3
dvar float+ produ [PPS_MATERIALS][horizon2]; //RR4
dvar float+ use[PPS_MATERIALS][horizon]; //RR5
dvar float+ sobra[PosProd] in 0..1;//RR7
dvar boolean sumaparcialsetup[PosSet];
dvar float auxiliar_target[PPS_MATERIALS][1..2];
dvar float+ available[PPS_MATERIALS][horizon];//RR6
dvar float+ hold_cost; //RR8
dvar float+ setup_cost; //RR11
dvar float+ back_cost; //RR10
dvar float+ prod_cost; //RR9
dvar float+ ingreso; //RR13
dvar float+ target_cost; //RR12
dexpr float total_benefit = ingreso - hold_cost - back_cost - target_cost
- prod_cost - setup_cost;
// Finalizar definición de variables de decisión ----------------------
// Función Objetivo ---------------------------------------------------
maximize total_benefit;
```

```
// Comienzo Restricciones ------------------------------------------------
subject to{
      forall (i in PPS_MATERIALS){
            stock[i][1]==available[i][1]-sales[i][1]-use[i][1]; //RR14
            backorder[i][1] + sales[i][1] == sum(h in
ppsDemand2[1]:h.material==i.material)h.demand; //RR16
            available [i][1] == produ[i][1-
leadtimes[mat_stage[i]]];//RR20
            use[i][1]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][1]*b.factor;//RR18
            produ[i][1]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, 1>]-sobra[<l, i.material,
1>])*n));//RR19
            forall (e in epochs,k in start[e]..start[e+1]-1:k>1){
                  stock[i][k]==stock[i][k-1]+available[i][k]-sales[i][k]-
use[i][k]; //RR15
                  backorder[i][k] + sales[i][k] == backorder[i][k-
1]+sum(h in ppsDemand2[e]:h.material==i.material)h.demand; //RR17
                  available [i][k] == produ[i][k-
leadtimes[mat_stage[i]]];//RR20
                  use[i][k]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][k]*b.factor;//RR18
                  produ[i][k]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, k>]-sobra[<l, i.material,
k>])*n));//RR19
            }
            forall (j in horizon2 : j <1){
                  produ[i][j]==0;
            }
      }

      forall (c in PosProd){
            sobra[c]<=mach_prod[c];//RR21
      }

// RR23': Restricción de capacidad de máquina --------------------------
forall ( e in epochs,i in ppsMachines2[1]){
                  dur_epochs[e]-sum(<i.machine,j,k> in PosProd:
k>=start[e] &&
k<start[e+1])(sobra[<i.machine,j,k>])<=machinecapacity[i][e];
}
// RR22: Evitar múltiples producciones y setup -------------------------
forall(i in ppsMachines2[1], k in horizon){
            sum(<i.machine, j, k> in PosProd)mach_prod[<i.machine,j,k>]+
sum(<i.machine, j, l, k> in PosSet) mach_setup[<i.machine, j, l, k>]==1;
}

// RR24: Restricción sobre SumaParcialSetup ----------------------------
forall (<m,i,j,l,o> in PPS_SETUP){
      forall (<m,i,j,k> in PosSet : k<=l+1){
            sumaparcialsetup[<m,i,j,k>]==0;
      }
      forall (<m,i,j,k> in PosSet:k>l+1){
               sumaparcialsetup[<m,i,j,k>]<=mach_prod[<m,i,k-l-1>];
                  forall (s in 1..l){
                  sumaparcialsetup[<m,i,j,k>]<=mach_setup[<m,i,j,k-s>];
                  }
```

```
      }
}

// RR25: Habilitación para el cambio de producción ----------------------
forall (k in horizon:k>1){
      forall (<m,j,k> in PosProd){
            mach_prod[<m,j,k>]<=mach_prod[<m,j,k-1>]+sum(<m,i,j,k> in
PosSet)sumaparcialsetup[<m,i,j,k>];
      }
}
// Comienza el cálculo de costos e ingreso ------------------------------
// RR26: Costo de Almacenamiento ----------------------------------------
hold_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)stock[i][k]*i.holdingcost+sum(i in PPS_MATERIALS)
stock[i][var_horizon]*0.5*i.holdingcost;
// RR27: Costo de Ruptura -----------------------------------------------
back_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)backorder[i][k]*i.backordercost+sum(i in PPS_MATERIALS)
backorder[i][var_horizon]*0.5*i.backordercost;
// RR30: Costo por Inventario Objetivo ----------------------------------
forall (i in PPS_MATERIALS){
      auxiliar_target[i][1]==i.target-
stock[i][var_horizon]+backorder[i][var_horizon];
      auxiliar_target[i][2]==0;
}
target_cost==sum(i in PPS_MATERIALS)max(p in 1..2)
auxiliar_target[i][p]*inv_deficit;
// RR28: Costo de Producción --------------------------------------------
prod_cost==sum(p in PPS_MATERIALPROD, <p.machine, p.material, k> in
PosProd) (p.prodcost*p.prodrate*(mach_prod[<p.machine, p.material, k>]-
sobra[<p.machine,p.material,k>]));
// RR29: Costo de Setup -------------------------------------------------
setup_cost == sum(s in PPS_SETUP, <s.machine, s.materialFrom,
s.materialTo, k> in PosSet) (s.setupcost*mach_setup[<s.machine,
s.materialFrom, s.materialTo, k>]);
// RR31: Ingreso por Ventas ---------------------------------------------
ingreso==sum(k in horizon, i in PPS_MATERIALS)sales[i][k]*i.price;
}
// Fin Restricciones ----------------------------------------------------
// FIN DE PROGRAMA------------------------------------------------------
```

9.5 Resolución con VPCPP combinadas: Módulo Inicial

```
/* DESCRIPCION GENERAL-------------------------------------------------
La idea de este módulo es determinar qué materiales producirá cada
máquina en cada época. Esta información se pasará al módulo final a
través del vector "demandaenarbol3". La información de entrada se
modifica para ajustarse a la naturaleza de un PAPP.
------------------------------------------------------------------------*/
// Definición de Tuples -----------------------------------------------
tuple TPpsBom {
  string material;
  string component;
  float factor;
};

tuple TPpsDemand {
  string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsDemand2 {
  key string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsLeadtimes {
  int stage1;
  int stage2;
  int leadtime;
};

tuple TPpsMachines {
  int stage;
  string machine;
  int epoch;
  float nethours;
};

tuple TPpsMachines2 {
  int stage;
  key string machine;
  int epoch;
  float nethours;
};

tuple TPpsMaterialprod {
  string machine;
  string material;
  float prodrate;
  float prodcost;
};
```

```
tuple TPpsMaterials {
  key string material;
  float price;
  float target;
  float holdingcost;
  float backordercost;
};

tuple TPpsParameters {
  key string parameterid;
  float parametervalue;
};
// Fin definición de tuples ------------------------------------------
// Cargar tuples de archivo .dat -------------------------------------
{TPpsBom} PPS_BOM = ...;
{TPpsDemand} PPS_DEMAND = ...;
{TPpsLeadtimes} PPS_LEADTIMES = ...;
{TPpsMachines} PPS_MACHINES = ...;
{TPpsMaterialprod} PPS_MATERIALPROD = ...;
{TPpsMaterials} PPS_MATERIALS = ...;
{TPpsParameters} PPS_PARAMETERS = ...;
// Comienzo creación de variables ------------------------------------
int num_epochs;
execute NUM_EPOCHS{
 for(var s in PPS_DEMAND){
      if(num_epochs<s.epoch){
            num_epochs=s.epoch;
      }
 }
}
range epochs=1..num_epochs;
//Dividir DEMANDA y CAPACIDAD según la época -------------------------
{TPpsDemand2} ppsDemand2[i in epochs] = {<a.material, a.epoch,
a.duration, a.demand>|a in PPS_DEMAND:a.epoch==i};
{TPpsMachines2} ppsMachines2[i in epochs] = {<a.stage, a.machine,
a.epoch,a.nethours>|a in PPS_MACHINES:a.epoch==i};
//Guardar la duración de las épocas en un vector --------------------
int dur_epochs[epochs];
execute DUR_EPOCHS {
      for (var e in epochs){
            for (var s in ppsDemand2[e]){
                  if (e == s.epoch){
                        dur_epochs[e]=s.duration;
                        break;
                  }
            }
      }
}
//Cargar parámetros en variables y rangos ---------------------------
int var_horizon;
int num_stages;
float inv_deficit;
execute PARAMETERS{
      var_horizon = PPS_PARAMETERS.find("Horizon").parametervalue;
      num_stages = PPS_PARAMETERS.find("Stages").parametervalue;
      inv_deficit = PPS_PARAMETERS.find("InvDeficitCost").parametervalue;
}
```

```
range horizon=1..var_horizon;
range stages = 1..num_stages;
//Cargar leadtimes en un vector ----------------------------------------
int leadtimes[stages];
execute LEADTIMES{
      leadtimes[num_stages]=0;
      for (var s in stages){
            for(var l in PPS_LEADTIMES){
                  if(l.stage1==s){
                        leadtimes[s]=l.leadtime;
                  }
            }
      }
}
//Determinar el máximo leadtime ----------------------------------------
int max_lead;
execute MAX_LEAD{
      for (var s in stages){
            if (max_lead<leadtimes[s]){
                  max_lead=leadtimes[s];
            }
      }
}
//Determinar la etapa productiva de cada producto ----------------------
int mat_stage[PPS_MATERIALS];
execute MAT_STAGE{
      for (var m in PPS_MATERIALS){
            mat_stage[m]=1;
            for (var p in PPS_MATERIALPROD){
                  if (m.material == p.material){

      mat_stage[m]=ppsMachines2[1].find(p.machine).stage;
                        }
                  }
            }
      }
//Definir el parámetro MixMachines y los vectores que se utilizarán en
//el post procesamiento de la solución (generación del ranking y
//traspaso de info) ----------------------------------------------------
int MixMachines=3; // Se cargó 3 como default
int demandaenarbol3[PPS_MATERIALS][epochs];
int demandaenarbol4[PPS_MATERIALS][epochs];
//Cargar el vector CAPACIDAD -------------------------------------------
float machinecapacity[ppsMachines2[1]][epochs];
execute MACHINECAPACITY{
for (var e in epochs){
      for (var a in ppsMachines2[e]){
      machinecapacity[ppsMachines2[1].find(a.machine)][e]=a.nethours;
      }
}
}
// Horizon2 es un rango que tiene algunos indices negativos
// Se utiliza para el vector 'Produ' para evitar problemas de índice
// negativo en el vector 'Disponible' ya que este último mira al vector
// 'Produ' lead time utems en el pasado.
range horizon2 = -max_lead..var_horizon;
//Cargar un vector con la utem de comienzo de cada época ---------------
```

```
int start[1..num_epochs+1];
execute START{
      start[num_epochs+1]=var_horizon+1;
      start[1]=1;
      for (var e in epochs){
            if (e>1){
            start[e] = start[e-1] + dur_epochs[e-1];
            }
      }
}
// Se define PosProd para explotar el esparcimiento --------------------
tuple prueba{
      string machine;
      string material1;
      int time;
}
setof(prueba) conjunto2[e in epochs]={<i,j,k> | <i,j,l,m> in
PPS_MATERIALPROD, k in start[e]..start[e+1]-1};
{prueba} PosProd = union (e in epochs) conjunto2[e];
// Comenzar la creación de variables de decisión ------------------------
dvar float mach_prod[PosProd] in 0..1;
dvar float+ stock [PPS_MATERIALS][horizon];
dvar float+ sales [PPS_MATERIALS][horizon];
dvar float+ backorder [PPS_MATERIALS][horizon];
dvar float+ produ [PPS_MATERIALS][horizon2];
dvar float+ use[PPS_MATERIALS][horizon];
dvar float+ available[PPS_MATERIALS][horizon];
dvar float+ hold_cost;
dvar float+ back_cost;
dvar float+ prod_cost;
dvar float+ target_cost;
dvar float+ epoch_revenue[PPS_MATERIALS][epochs];
dvar float+ tot_revenue;
dexpr float total_benefit = tot_revenue - hold_cost - back_cost-
prod_cost;
// Finalizar definición de variables de decisión ------------------------
// Función Objetivo -----------------------------------------------------
maximize total_benefit;
// Comienzo Restricciones -----------------------------------------------
subject to{
      forall (i in PPS_MATERIALS){
            stock[i][1]==available[i][1]-sales[i][1]-use[i][1];
            backorder[i][1] + sales[i][1] == sum(h in
ppsDemand2[1]:h.material==i.material)h.demand;
            available [i][1] == produ[i][1-leadtimes[mat_stage[i]]];
            use[i][1]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][1]*b.factor;
            produ[i][1]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, 1>])*n));
            forall (e in epochs,k in start[e]..start[e+1]-1:k>1){
                  stock[i][k]==stock[i][k-1]+available[i][k]-sales[i][k]-
use[i][k];
                  backorder[i][k] + sales[i][k] == backorder[i][k-
1]+sum(h in ppsDemand2[e]:h.material==i.material)h.demand;
                  available [i][k] == produ[i][k-
leadtimes[mat_stage[i]]];
```

```
                  use[i][k]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][k]*b.factor;
                  produ[i][k]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, k>])*n));
            }
            forall (j in horizon2 : j <1){
                  produ[i][j]==0;
            }
      }

      forall (e in epochs){
            forall(i in ppsMachines2[1], k in start[e]..start[e+1]-1){
                  sum(<i.machine, j, k> in
PosProd)mach_prod[<i.machine,j,k>]<=1;
            }
      }

      forall (e in epochs, i in ppsMachines2[1]){
            sum(<i.machine, j, k> in PosProd:k>=start[e] &&
k<start[e+1])mach_prod[<i.machine,j,k>]<=machinecapacity[i][e];
      }
  // Comienzo cálculo de costos e ingreso ------------------------------
      hold_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)stock[i][k]*i.holdingcost+sum(i in PPS_MATERIALS)
stock[i][var_horizon]*0.5*i.holdingcost;
      back_cost==sum(k in horizon : k<var_horizon, i in
PPS_MATERIALS)backorder[i][k]*i.backordercost+sum(i in PPS_MATERIALS)
backorder[i][var_horizon]*0.5*i.backordercost;
      prod_cost==sum(p in PPS_MATERIALPROD, <p.machine, p.material, k> in
PosProd) (p.prodcost*p.prodrate*(mach_prod[<p.machine, p.material, k>]));

      target_cost==sum(i in PPS_MATERIALS)
backorder[i][var_horizon]*inv_deficit;
      forall (e in epochs, i in PPS_MATERIALS){
            epoch_revenue[i][e]==sum(k in horizon:k>=start[e] &&
k<start[e+1])(sales[i][k]*i.price);
      }
      tot_revenue==sum(e in epochs, i in
PPS_MATERIALS)epoch_revenue[i][e];
}
// Fin Restricciones ---------------------------------------------------
// Fin Optimización ----------------------------------------------------
// Comienzo Post procesamiento -----------------------------------------
// IMPORTANTE: En este bloque de postprocesamiento se carga el vector
//"demandaenarbol3".Este vector guarda la información a pasar al otro
//módulo de optimización. El vector "demandaenarbol4" funciona como un
//auxiliar ya que para cada producto introducido en "demandaenarbol3"
//verifica si alguno de sus componentes viola el límite "MixMachines".
// ---------------------------------------------------------------------
execute DEMANDAENARBOL3{
// Determinar máximo global
var maxtotal;
var maxtrabajo;
maxtotal=0;
for (var eta in epochs){
      if (eta < num_epochs){
      for (var mate in PPS_MATERIALS){
```

```
            if
(epoch_revenue[mate][eta]+epoch_revenue[mate][eta+1]>maxtotal){

      maxtotal=epoch_revenue[mate][eta]+epoch_revenue[mate][eta+1];
            }
      }
      }
}

var maxant;
var max1;
var trabajo = new Array();
for (var e in epochs){
      maxant = maxtotal + 1;
      max1 = 0;
      for (var a in PPS_MATERIALS){
            // Determinar siguiente máximo
            for (var b in PPS_MATERIALS){
                  if (e < num_epochs){
                  if (epoch_revenue[b][e]+epoch_revenue[b][e+1]>max1 &&
epoch_revenue[b][e]+epoch_revenue[b][e+1]<maxant){
                        max1= epoch_revenue[b][e]+epoch_revenue[b][e+1];
                  }
                  }
                  else {
                  if (epoch_revenue[b][e]>max1 &&
epoch_revenue[b][e]<maxant){
                        max1= epoch_revenue[b][e];
                  }
                  }
            }
            if (max1 == 0){
            break; // Si el siguiente máximo es cero, salir de la época
            }
            // Actualizar demandaenarbol4
            for (var c in PPS_MATERIALS){
                  if ((e == num_epochs && epoch_revenue[c][e] == max1) ||
(e < num_epochs && epoch_revenue[c][e]+epoch_revenue[c][e+1] == max1)){
                        demandaenarbol3[c][e]=1;
                        for (var reseteomater in PPS_MATERIALS){
                              for(var reseteoepochs in epochs){

      demandaenarbol4[reseteomater][reseteoepochs]=0;
                              }
                        }
                        for (var et in epochs){
                              for (var ch in stages){
                                    for (var ab in PPS_MATERIALS){
                                          if(mat_stage[ab]==num_stages-
ch+1 && ch==1){
                                                if
(demandaenarbol3[ab][et]==1){
                                                demandaenarbol4[ab][et]=1;
                                                }
                                          }
                                          else{
                                          if (demandaenarbol3[ab][et]==1){
```

```
                                                demandaenarbol4[ab][et]=1;
                                                }
                                                if (demandaenarbol4[ab][et]==0){
                                                      for (var bb in PPS_BOM){

      if(bb.component==ab.material &&
demandaenarbol4[PPS_MATERIALS.find(bb.material)][et]>0 && bb.factor>0){

      demandaenarbol4[ab][et]=1;
                                                                  break;
                                                                  }
                                                            }
                                                }
                                                }
                                          }
                                    }
                              }
                        for (var mach in ppsMachines2[1]){
            trabajo[mach.machine]=0;
                        }
            for (var u in PPS_MATERIALPROD){

      trabajo[u.machine]=trabajo[u.machine]+demandaenarbol4[PPS_MATERIALS
.find(u.material)][e];
            }
            maxtrabajo=0;
            for (var machines in ppsMachines2[1]){
                  if (trabajo[machines.machine]>maxtrabajo){
                        maxtrabajo=trabajo[machines.machine];
                  }
            }
            // Si se viola a MixMachines, no incluir el producto en
            demandaenarbol3
            if (maxtrabajo>MixMachines){
                  demandaenarbol3[c][e]=0;
            }
                  }
            }
            maxant=max1;
            max1=0;
      }
}
}
// Fin PostProcesamiento ------------------------------------------------
// FIN DE PROGRAMA ------------------------------------------------------
```

9.6 Resolución con VPCPP combinadas: Módulo Final

```
/* ACLARACIÓN----------------------------------------------------------
Este módulo tiene dos inputs: la información original que se cargan en
tuples y la información del módulo INICIAL que se carga en el vector
"demandaenarbol3".
-----------------------------------------------------------------------*/
// Definición de Tuples ----------------------------------------------
tuple TPpsBom {
  string material;
  string component;
  float factor;
};

tuple TPpsDemand {
  string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsDemand2 {
  key string material;
  int epoch;
  int duration;
  float demand;
};

tuple TPpsLeadtimes {
  int stage1;
  int stage2;
  int leadtime;
};

tuple TPpsMachines {
  int stage;
  string machine;
  int epoch;
  float nethours;
};

tuple TPpsMachines2 {
  int stage;
  key string machine;
  int epoch;
  float nethours;
};

tuple TPpsMaterialprod {
  string machine;
  string material;
  float prodrate;
  float prodcost;
};

tuple TPpsMaterials {
```

```
  key string material;
  float price;
  float target;
  float holdingcost;
  float backordercost;
};

tuple TPpsParameters {
  key string parameterid;
  float parametervalue;
};

tuple TPpsSetup {
  string machine;
  string materialFrom;
  string materialTo;
  int setuptime;
  float setupcost;
};
// Fin definición de tuples ------------------------------------------
// Cargar tuples de archivo .dat -------------------------------------
{TPpsBom} PPS_BOM = ...;
{TPpsDemand} PPS_DEMAND = ...;
{TPpsLeadtimes} PPS_LEADTIMES = ...;
{TPpsMachines} PPS_MACHINES = ...;
{TPpsMaterialprod} PPS_MATERIALPROD = ...;
{TPpsMaterials} PPS_MATERIALS = ...;
{TPpsParameters} PPS_PARAMETERS = ...;
{TPpsSetup} PPS_SETUP = ...;
// Comienzo creación de variables ------------------------------------
int num_epochs;
execute NUM_EPOCHS{
 for(var s in PPS_DEMAND){
      if(num_epochs<s.epoch){
            num_epochs=s.epoch;
      }
 }
}
range epochs=1..num_epochs;
//Dividir DEMANDA y CAPACIDAD según la época -------------------------
{TPpsDemand2} ppsDemand2[i in epochs] = {<a.material, a.epoch,
a.duration, a.demand>|a in PPS_DEMAND:a.epoch==i};
{TPpsMachines2} ppsMachines2[i in epochs] = {<a.stage, a.machine,
a.epoch,a.nethours>|a in PPS_MACHINES:a.epoch==i};
//-------------------------------------------------------------------
//Guardar la duración de las épocas en un vector --------------------
int dur_epochs[epochs];
execute DUR_EPOCHS {
      for (var e in epochs){
            for (var s in ppsDemand2[e]){
                  if (e == s.epoch){
                        dur_epochs[e]=s.duration;
                        break;
                  }
            }
      }
}
```

```
//Cargar parámetros en variables y rangos ------------------------------
int var_horizon;
int num_stages;
float inv_deficit;
execute PARAMETERS{
      var_horizon = PPS_PARAMETERS.find("Horizon").parametervalue;
      num_stages = PPS_PARAMETERS.find("Stages").parametervalue;
      inv_deficit = PPS_PARAMETERS.find("InvDeficitCost").parametervalue;
}
range horizon=1..var_horizon;
range stages = 1..num_stages;
//-----------------------------------------------------------------------
// Definir checkstep: período de revisión variable (VPCPP 1) ------------
// Definir setstep: horas pautadas para cambio de producción (VPCPP 2) --
// Se introdujeron valores razonables por default ----------------------
int setstep[epochs];
int checkstep;
execute STEPS{
      for (var e in epochs){
            setstep[e]=Opl.ceil(0.05*dur_epochs[e]);
      }
      checkstep=Opl.ceil(0.01*var_horizon);
}
//Cargar leadtimes en un vector ----------------------------------------
int leadtimes[stages];
execute LEADTIMES{
      var max;
      for (var s in stages){
            max = 0;
            for(var l in PPS_LEADTIMES){
                  if(l.stage1==s && l.leadtime>max){
                        max=l.leadtime;
                  }
            }
            leadtimes[s]=max;
      }
      leadtimes[num_stages]=0;
}
//Determinar el máximo leadtime ----------------------------------------
int max_lead;
execute MAX_LEAD{
      for (var s in stages){
            if (max_lead<leadtimes[s]){
                  max_lead=leadtimes[s];
            }
      }
}
//Determinar la etapa productiva de cada producto ----------------------
int mat_stage[PPS_MATERIALS];
execute MAT_STAGE{
      for (var m in PPS_MATERIALS){
      mat_stage[m]=1;
            for (var p in PPS_MATERIALPROD){
                  if (m.material == p.material){

      mat_stage[m]=ppsMachines2[1].find(p.machine).stage;
                  }
```

```
                  }
            }
      }
// Demandaenarbol3 se carga del modulo INICIAL -------------------------
float demandaenarbol3[PPS_MATERIALS][epochs];
// Demandaenarbol4 tiene la misma utilidad que en módulo INICIAL --------
float demandaenarbol4[PPS_MATERIALS][epochs];
//Construir "demandaenarbol4" a partir de "demandaenarbol3" usando la
info del BOM ---------------------------------------------------------
execute DEMANDAENARBOL4{
for (var e in epochs){
      for (var c in stages){
            for (var a in PPS_MATERIALS){
                  if(mat_stage[a]==num_stages-c+1 && c==1){
                        if (demandaenarbol3[a][e]==1){
                              demandaenarbol4[a][e]=1;
                        }
                  }
                  else{
                        if (demandaenarbol3[a][e]==1){
                              demandaenarbol4[a][e]=1;
                        }
                        if (demandaenarbol4[a][e]==0){
                              for (var b in PPS_BOM){
                                    if(b.component==a.material &&
demandaenarbol4[PPS_MATERIALS.find(b.material)][e]>0 && b.factor>0){
                                    demandaenarbol4[a][e]=1;
                                    break;
                                    }
                              }
                        }
                  }
            }
      }
}
}
int works[ppsMachines2[1]][epochs]; //vector que guarda la cantidad de
productos que cada máquina manipulará en cada época
execute{
for (var t in epochs){
      for (var mach in ppsMachines2[1]){
      works[mach][t]=0;
      }
    for (var u in PPS_MATERIALPROD){

works[ppsMachines2[1].find(u.machine)][t]=works[ppsMachines2[1].find(u.ma
chine)][t]+demandaenarbol4[PPS_MATERIALS.find(u.material)][t];
    }
}

}
// Costo promedio de la hora de setup por maquina (VPCPP 3)-------------
float promsetup[ppsMachines2[1]];
execute{
for (var m in ppsMachines2[1]){
      var cant = 0;
      var suma = 0;
```

```
      for (var s in PPS_SETUP){
      if (m.machine == s.machine && s.setupcost>0){
            cant = cant + 1;
            suma = suma + s.setupcost;
      }
      }
      promsetup[m]=suma/cant;
}
}
// Construir "hasdemand": Indicador para aplicar VPCPP 1 (Parte b)-------
float hasdemand[PPS_MATERIALS][epochs];
execute HASDEMAND{
for (var e in epochs){
      if (e==1){
            for (var a in PPS_MATERIALS){
                        if (ppsDemand2[e].find(a.material).demand>0){
                              hasdemand[a][e]=1;
                        }
                        else{
                        hasdemand[a][e]=0;
                  }
}
      }
      else{
      for (var b in PPS_MATERIALS){
                        if (ppsDemand2[e].find(b.material).demand>0 ||
hasdemand[b][e-1]==1){
                              hasdemand[b][e]=1;
                        }
                        else{
                        hasdemand[b][e]=0;
                  }
      }
      }

}
}
//Cargar el vector CAPACIDAD ----------------------------------------------
float machinecapacity[ppsMachines2[1]][epochs];
execute MACHINECAPACITY{
for (var e in epochs){
      for (var a in ppsMachines2[e]){

      machinecapacity[ppsMachines2[1].find(a.machine)][e]=a.nethours;
      }
}
}
// Los siguientes conjuntos utilizan la información de "demandaenarbol4"
// y de "hasdemand" para explotar el esparcimiento del problema ---------
{TPpsMaterials} ppsMaterialsfree[e in epochs] = {a| a in
PPS_MATERIALS:demandaenarbol4[a][e]>=1};
{TPpsMaterials} ppsMaterialsdemand[e in epochs] = {a| a in
PPS_MATERIALS:hasdemand[a][e]==1};
{TPpsSetup} ppsSetupfree[e in epochs] = {a| a in
PPS_SETUP:demandaenarbol4[<a.materialFrom>][e]>=1 &&
demandaenarbol4[<a.materialTo>][e]>=1 && a.materialFrom!=a.materialTo};
//-------------------------------------------------------------------------
```

```
// Horizon2 es un rango que tiene algunos indices negativos
// Se utiliza para el vector 'Produ' para evitar problemas de índice
// negativo en el vector 'Disponible' ya que este último mira al vector
// 'Produ' lead time utems en el pasado.
range horizon2 = -max_lead..var_horizon;
//Cargar un vector con la utem de comienzo de cada época----------------
int start[1..num_epochs+1];
execute START{
      start[num_epochs+1]=var_horizon+1;
      start[1]=1;
      for (var e in epochs){
            if (e>1){
            start[e] = start[e-1] + dur_epochs[e-1];
            }
      }
}
// Se definen PosProd y PosSet para explotar el esparcimiento ----------
tuple seteo{
      string machine;
      string materialFrom;
      string materialTo;
      int time;
}
tuple prueba{
      string machine;
      string material1;
      int time;
}

setof(prueba) conjunto2[e in epochs]={<i,j,k> | <j,a,b,c,d> in
ppsMaterialsfree[e], <i,j,l,m> in PPS_MATERIALPROD, k in
start[e]..start[e+1]-1};
{prueba} PosProd=union (e in epochs) conjunto2[e];
setof(seteo) conjuntoseteo2[e in epochs]={<i,j,l,k> | <i,j,l,m,n> in
ppsSetupfree[e], k in start[e]+1..start[e+1]-1:k mod setstep[e] ==
start[e] mod setstep[e]};
{TPpsSetup} ppsSetupfreeUnion[e in epochs] = {a| a in
PPS_SETUP:demandaenarbol4[<a.materialFrom>][e-1]>=1 &&
demandaenarbol4[<a.materialTo>][e]>=1 && e>1 &&
a.materialFrom!=a.materialTo};
setof(seteo) conjuntoseteounion[e in epochs]={<i,j,l,start[e]> |
<i,j,l,m,n> in ppsSetupfreeUnion[e]};
{seteo} conjuntoseteo3=union (e in epochs) conjuntoseteo2[e];
{seteo} conjuntoseteounion3= union (e in epochs) conjuntoseteounion[e];
{seteo} PosSet = conjuntoseteo3 union conjuntoseteounion3;
// Comenzar la creación de variables de decisión ------------------------
dvar boolean mach_prod[PosProd];
dvar boolean mach_setup[ppsMachines2[1]][horizon];
dvar float+ stock [PPS_MATERIALS][horizon];
dvar float+ sales [PPS_MATERIALS][horizon];
dvar float+ backorder [PPS_MATERIALS][horizon];
dvar float+ produ [PPS_MATERIALS][horizon2];
dvar float+ use[PPS_MATERIALS][horizon];
dvar float+ sobra[PosProd] in 0..1;
dvar boolean sumaparcialsetup[PosSet];
dvar float auxiliar_target[PPS_MATERIALS][1..2];
dvar float+ available[PPS_MATERIALS][horizon];
```

```
dvar float+ hold_cost;
dvar float+ back_cost_etapas[epochs];
dexpr float back_cost = sum(e in epochs)back_cost_etapas[e];
dvar float+ prod_cost;
dvar float+ epoch_revenue[epochs];
dexpr float tot_revenue = sum(e in epochs)epoch_revenue[e];
dvar float+ target_cost;
dvar float+ setup_cost;
dexpr float total_benefit = tot_revenue - hold_cost - back_cost -
target_cost - prod_cost - setup_cost;
// Finalizar definición de variables de decisión ----------------------
// Función Objetivo ---------------------------------------------------
maximize total_benefit;
// Comienzo Restricciones ---------------------------------------------
subject to{
      forall (i in PPS_MATERIALS){
            stock[i][1]==available[i][1]-sales[i][1]-use[i][1];
            backorder[i][1] + sales[i][1] == sum(h in
ppsDemand2[1]:h.material==i.material)h.demand;
            available [i][1] == produ[i][1-leadtimes[mat_stage[i]]];
            use[i][1]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][1]*b.factor;
            if (i in ppsMaterialsfree[1]){
            produ[i][1]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, 1>]-sobra[<l, i.material,
1>])*n));
            }
            else{
            produ[i][1]==0;
            }
            forall (e in epochs,k in start[e]..start[e+1]-1:k>1){
                  stock[i][k]==stock[i][k-1]+available[i][k]-sales[i][k]-
use[i][k];
                  backorder[i][k] + sales[i][k] == backorder[i][k-
1]+sum(h in ppsDemand2[e]:h.material==i.material)h.demand;
                  available [i][k] == produ[i][k-
leadtimes[mat_stage[i]]];
                  use[i][k]==sum(b in
PPS_BOM:b.component==i.material)produ[<b.material>][k]*b.factor;
                  if (i in ppsMaterialsfree[e]){
                  produ[i][k]==(sum(<l, i.material, n,m> in
PPS_MATERIALPROD)((mach_prod[<l, i.material, k>]-sobra[<l, i.material,
k>])*n));
                  }
                  else{
                  produ[i][k]==0;
                  }
            }
            forall (j in horizon2 : j <1){
                  produ[i][j]==0;
            }
      }

      forall (c in PosProd){
            sobra[c]<=mach_prod[c];
      }
```

```
// Capacidad de máquinas ------------------------------------------------
      forall ( e in epochs,i in ppsMachines2[1]: works[i][e]>0){
                  dur_epochs[e]-sum(<i.machine,j,k> in
conjunto2[e])(sobra[<i.machine,j,k>])<=machinecapacity[i][e];
      }
// Evitar múltiples producciones y setups -------------------------------
      forall(i in ppsMachines2[1], k in horizon){
            sum(<i.machine, j, k> in PosProd)mach_prod[<i.machine,j,k>]+
mach_setup[i][k]==1;
      }
// Restricciones relacionadas con Setup ---------------------------------
// Condiciones sobre sumaparcialsetup -----------------------------------
      forall (e in epochs){
            forall (<m,i,j,l,o> in ppsSetupfree[e]){
                  forall (<m,i,j,k> in PosSet : k>start[e] &&
k<=l+start[e]){
                        sumaparcialsetup[<m,i,j,k>]==0;
                  }
                  forall (<m,i,j,k> in PosSet:k>l+start[e] &&
k<start[e+1]){
                           sumaparcialsetup[<m,i,j,k>]<=mach_prod[<m,i,k-
l-1>];
                             forall (s in 1..l){

      sumaparcialsetup[<m,i,j,k>]<=mach_setup[<m>][k-s];
                             }
                  }
            }
      }

      forall (e in epochs){
            forall (<m,i,j,l,o> in ppsSetupfreeUnion[e]){

sumaparcialsetup[<m,i,j,start[e]>]<=mach_prod[<m,i,start[e]-1-1>];
                             forall (s in 1..l){

      sumaparcialsetup[<m,i,j,start[e]>]<=mach_setup[<m>][start[e]-s];
                  }
            }
      }
// Fin condiciones sobre sumaparcialsetup -------------------------------
// Continuidad en la producción -----------------------------------------
      forall (e in epochs){
            forall (<m,j,k> in PosProd:k>start[e]&&k<start[e+1] && k mod
setstep[e] == start[e] mod setstep[e]){
                  mach_prod[<m,j,k>]<=mach_prod[<m,j,k-1>]+sum(<m,i,j,k>
in PosSet)sumaparcialsetup[<m,i,j,k>];
            }
      }
      forall (e in epochs){
            forall (<m,j,k> in PosProd:k>start[e]&&k<start[e+1] && k mod
setstep[e] != start[e] mod setstep[e]){
                  mach_prod[<m,j,k>]<=mach_prod[<m,j,k-1>];
            }
      }
// Continuidad en la producción para la primera hora de cada época ------
      forall (e in epochs: e >1){
```

```
            forall (<m,j,start[e]> in PosProd:<m,j,start[e]-1> in PosProd
&& works[<m>][e-1]>0 && machinecapacity[<m>][e-1]>0){
                  mach_prod[<m,j,start[e]>]<=mach_prod[<m,j,start[e]-
1>]+sum(<m,i,j,start[e]> in
conjuntoseteounion[e])sumaparcialsetup[<m,i,j,start[e]>];
            }
            forall (<m,j,start[e]> in PosProd:<m,j,start[e]-1> not in
PosProd && works[<m>][e-1]>0 && machinecapacity[<m>][e-1]>0){
                  mach_prod[<m,j,start[e]>]<=sum(<m,i,j,start[e]> in
conjuntoseteounion[e])sumaparcialsetup[<m,i,j,start[e]>];
            }
      }
// Fin restricciones relacionadas con setup ----------------------------
// Cálculo de costos e ingreso -----------------------------------------
      hold_cost==sum(k in horizon : k mod checkstep == 0, i in
PPS_MATERIALS)stock[i][k]*i.holdingcost*checkstep;
      forall (e in epochs){
            back_cost_etapas[e]==sum(k in horizon : k>=start[e] &&
k<start[e+1] && k mod checkstep==0, i in
ppsMaterialsdemand[e])backorder[i][k]*i.backordercost*checkstep;
      }
      forall (i in PPS_MATERIALS){
            auxiliar_target[i][1]==i.target-
stock[i][var_horizon]+backorder[i][var_horizon];
            auxiliar_target[i][2]==0;
      }
      target_cost==sum(i in PPS_MATERIALS)max(p in
1..2)auxiliar_target[i][p]*inv_deficit;
      prod_cost==sum(p in PPS_MATERIALPROD, <p.machine, p.material, k> in
PosProd) (p.prodcost*p.prodrate*(mach_prod[<p.machine, p.material, k>]-
sobra[<p.machine, p.material, k>]));
    setup_cost == sum(i in ppsMachines2[1], k in horizon)
(promsetup[i]*mach_setup[i][k]);
    forall (e in epochs){
            epoch_revenue[e]==sum(k in horizon : k>=start[e] &&
k<start[e+1], i in ppsMaterialsdemand[e])sales[i][k]*i.price;
      }
}
// Fin Restricciones ---------------------------------------------------
// FIN DE PROGRAMA -----------------------------------------------------
```

10 APÉNDICE B: DATOS DE ENTRADA

10.1 Datos de entrada del PAPP

```
PPS_BOM = {
<"A-2" "A-1" 2.2 >
<"B-2" "A-1" 1.3 >
<"B-2" "B-1" 1.4 >
<"C-2" "B-1" 1.1 >
<"D-2" "A-1" 1.6 >
<"E-2" "A-1" 3.2 >
<"F-2" "B-1" 1.7 >
};
PPS_DEMAND = {
<"A-2" 200 6.5 >
<"B-2" 200 29.3 >
<"C-2" 200 6.2 >
<"D-2" 200 23.7 >
<"E-2" 200 4.1 >
<"F-2" 200 16.6 >
};
PPS_MACHINES = {
<1 "S1MAQ1" 180 >
<2 "S2MAQ1" 180 >
<2 "S2MAQ2" 180 >
};
PPS_MATERIALPROD = {
<"S1MAQ1" "A-1" 56.2 3.87 >
<"S2MAQ1" "A-2" 18.3 4.9 >
<"S2MAQ1" "B-2" 25.1 5.06 >
<"S2MAQ2" "B-2" 1.6 4.3 >
<"S2MAQ2" "C-2" 10.3 2.2 >
<"S2MAQ1" "D-2" 20.7 4 >
<"S2MAQ2" "D-2" 17.1 4 >
<"S2MAQ1" "E-2" 9.2 10 >
<"S2MAQ2" "F-2" 32.2 7 >
<"S1MAQ1" "B-1" 24.6 6 >
};
PPS_MATERIALS = {
<"A-2" 40 0.0006 0.0024 >
<"B-2" 35 0.0006 0.0028 >
<"C-2" 50 0.0006 0.003 >
<"D-2" 20 0.0006 0.003 >
<"E-2" 50 0.0006 0.0032 >
<"F-2" 70 0.0006 0.0025 >
<"A-1" 0 0 0 >
<"B-1" 0 0 0 >
};
PPS_PARAMETERS = {
<"Horizon" 200 >
<"Stages" 2 >
};
```

10.2 Datos de entrada del PAPP multi-época

```
PPS_BOM = {
<"A-2" "A-1" 2 >
<"B-2" "A-1" 1 >
<"B-2" "B-1" 1.5 >
<"C-2" "B-1" 1.8 >
<"D-2" "A-1" 1 >
<"E-2" "A-1" 3 >
<"F-2" "B-1" 0.1 >
};
PPS_DEMAND = {
<"A-2" 1 50 15 >
<"A-2" 2 50 0 >
<"A-2" 3 50 0 >
<"A-2" 4 50 0 >
<"B-2" 1 50 30 >
<"B-2" 2 50 0 >
<"B-2" 3 50 0 >
<"B-2" 4 50 30 >
<"C-2" 1 50 10 >
<"C-2" 2 50 12 >
<"C-2" 3 50 25 >
<"C-2" 4 50 20 >
<"D-2" 1 50 12 >
<"D-2" 2 50 0 >
<"D-2" 3 50 12 >
<"D-2" 4 50 0 >
<"E-2" 1 50 10 >
<"E-2" 2 50 0 >
<"E-2" 3 50 15 >
<"E-2" 4 50 20 >
<"F-2" 1 50 0 >
<"F-2" 2 50 0 >
<"F-2" 3 50 0 >
<"F-2" 4 50 70 >
};

PPS_MACHINES = {
<1 "S1MAQ1" 1 45 >
<1 "S1MAQ1" 2 50 >
<1 "S1MAQ1" 3 50 >
<1 "S1MAQ1" 4 50 >
<2 "S2MAQ1" 1 50 >
<2 "S2MAQ1" 2 40 >
<2 "S2MAQ1" 3 50 >
<2 "S2MAQ1" 4 20 >
<2 "S2MAQ2" 1 50 >
<2 "S2MAQ2" 2 50 >
<2 "S2MAQ2" 3 50 >
<2 "S2MAQ2" 4 10 >
};
PPS_MATERIALPROD = {
<"S1MAQ1" "A-1" 560 3.87 >
<"S2MAQ1" "A-2" 60.75 4.9 >
<"S2MAQ1" "B-2" 76.5 5.06 >
<"S2MAQ2" "B-2" 69.75 4.3 >
<"S2MAQ2" "C-2" 56.25 2.2 >
<"S2MAQ1" "D-2" 60.75 4 >
<"S2MAQ2" "D-2" 63 4 >
<"S2MAQ1" "E-2" 90 10 >
<"S2MAQ2" "F-2" 83.25 7 >
<"S1MAQ1" "B-1" 240 6 >
};
PPS_MATERIALS = {
<"A-2" 40 0.6 0.024 >
<"B-2" 35 0.6 0.028 >
<"C-2" 50 0.6 0.03 >
<"D-2" 20 0.6 0.03 >
<"E-2" 50 0.6 0.032 >
<"F-2" 70 0.6 0.025 >
<"A-1" 0 0 0 >
<"B-1" 0 0 0 >
};
PPS_PARAMETERS = {
<"Horizon" 200 >
<"Stages" 2 >
};
```

10.3 Datos de entrada del PCPP

```
PPS_BOM = {
<"A-2" "A-1" 2 >
<"B-2" "A-1" 1 >
<"B-2" "B-1" 1.5 >
<"C-2" "B-1" 1.8 >
<"D-2" "A-1" 1 >
<"E-2" "A-1" 3 >
<"F-2" "B-1" 0.1 >
};
PPS_DEMAND = {
<"A-2" 1 7 15 >
<"A-2" 2 7 0 >
<"A-2" 3 7 0 >
<"A-2" 4 7 0 >
<"B-2" 1 7 30 >
<"B-2" 2 7 0 >
<"B-2" 3 7 0 >
<"B-2" 4 7 30 >
<"C-2" 1 7 10 >
<"C-2" 2 7 12 >
<"C-2" 3 7 25 >
<"C-2" 4 7 20 >
<"D-2" 1 7 12 >
<"D-2" 2 7 0 >
<"D-2" 3 7 12 >
<"D-2" 4 7 0 >
<"E-2" 1 7 10 >
<"E-2" 2 7 0 >
<"E-2" 3 7 15 >
<"E-2" 4 7 20 >
<"F-2" 1 7 0 >
<"F-2" 2 7 0 >
<"F-2" 3 7 0 >
<"F-2" 4 7 70 >
};
PPS_LEADTIMES = {
<1 2 1 >};
PPS_MACHINES = {
<1 "S1MAQ1" 1 7 >
<1 "S1MAQ1" 2 6 >
<1 "S1MAQ1" 3 5 >
<1 "S1MAQ1" 4 4 >
<2 "S2MAQ1" 1 7 >
<2 "S2MAQ1" 2 7 >
<2 "S2MAQ1" 3 7 >
<2 "S2MAQ1" 4 7 >
<2 "S2MAQ2" 1 7 >
<2 "S2MAQ2" 2 7 >
<2 "S2MAQ2" 3 7 >
<2 "S2MAQ2" 4 3 >
};
PPS_MATERIALPROD = {
<"S1MAQ1" "A-1" 560 3.87 >
<"S2MAQ1" "A-2" 60.75 4.9 >
<"S2MAQ1" "B-2" 76.5 5.06 >
<"S2MAQ2" "B-2" 69.75 4.3 >
<"S2MAQ2" "C-2" 56.25 2.2 >
<"S2MAQ1" "D-2" 60.75 4 >
<"S2MAQ2" "D-2" 63 4 >
<"S2MAQ1" "E-2" 90 10 >
<"S2MAQ2" "F-2" 83.25 7 >
<"S1MAQ1" "B-1" 240 6 >
};
PPS_MATERIALS = {
<"A-2" 40 5 0.06 0.24 >
<"B-2" 35 20 0.06 0.28 >
<"C-2" 50 30 0.06 0.3 >
<"D-2" 20 10 0.06 0.3 >
<"E-2" 50 4 0.06 0.32 >
<"F-2" 70 10 0.06 0.25 >
<"A-1" 0 0 0 0 >
<"B-1" 0 0 0 0 >
};
PPS_PARAMETERS = {
<"Horizon" 28 >
<"InvDeficitCost" 0.8 >
<"Stages" 2 >
};
PPS_SETUP = {
<"S1MAQ1" "A-1" "B-1" 2 135.751 >
<"S1MAQ1" "B-1" "A-1" 3 100 >
<"S2MAQ1" "A-2" "B-2" 5 265.524 >
<"S2MAQ1" "A-2" "D-2" 2 135.751 >
<"S2MAQ1" "A-2" "E-2" 3 135.751 >
<"S2MAQ1" "B-2" "A-2" 5 162.901 >
<"S2MAQ1" "B-2" "D-2" 3 135.751 >
<"S2MAQ1" "B-2" "E-2" 5 135.751 >
<"S2MAQ1" "D-2" "A-2" 2 135.751 >
<"S2MAQ1" "D-2" "B-2" 3 135.751 >
<"S2MAQ1" "D-2" "E-2" 1 135.751 >
<"S2MAQ1" "E-2" "A-2" 3 135.751 >
<"S2MAQ1" "E-2" "B-2" 5 135.751 >
<"S2MAQ1" "E-2" "D-2" 1 135.751 >
<"S2MAQ2" "B-2" "C-2" 3 135.751 >
<"S2MAQ2" "B-2" "D-2" 4 181.001 >
<"S2MAQ2" "B-2" "F-2" 5 162.901 >
<"S2MAQ2" "C-2" "B-2" 3 135.751 >
<"S2MAQ2" "C-2" "D-2" 1 135.751 >
<"S2MAQ2" "C-2" "F-2" 4 181.001 >
<"S2MAQ2" "D-2" "C-2" 1 135.751 >
<"S2MAQ2" "D-2" "B-2" 4 181.001 >
<"S2MAQ2" "D-2" "F-2" 2 135.751 >
<"S2MAQ2" "F-2" "C-2" 4 181.001 >
<"S2MAQ2" "F-2" "D-2" 2 135.751 >
<"S2MAQ2" "F-2" "B-2" 5 162.901 >
};
```