UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en ingeniería en Organización Industrial

PROCEDIMIENTO DE GENERACIÓN DE COLUMNAS. APLICACIONES A PROBLEMAS LOGÍSTICOS.

Autor:

González Morales, Miguel Ángel.

Tutor:

Sáez Aguado, Jesús

Departamento de Estadística e
Investigación Operativa.

Valladolid, Julio 2014.

AGRADECIMIENTOS

En primer lugar quisiera agradecer a Jesús Sáez Aguado la oportunidad que me ha brindado para realizar este proyecto y aprender de él, así como a la universidad de Valladolid por permitirme realizarlo.

A mis padres, con los que he compartido mis disgustos, pero también mis alegrías, y que han estado apoyándome desde el principio, animándome siempre a continuar hacia delante. A mis hermanos Sonia y José, y mis sobrinos Hugo y Lucía, así como al resto de mi familia que me ha apoyado y ha confiado en mí.

A todos mis profesores, desde el colegio hasta la universidad, en especial a todos aquellos que se han preocupado por mi aprendizaje, y me han permitido llegar hasta aquí. Por todo lo que he aprendido de vosotros, gracias.

A todos mis compañeros y amigos surgidos en la universidad, de los que también se aprende, y mucho. Me alegro haber compartido estos años con vosotros. Momentos de trabajos y estudio, momentos de intercambio de información, momentos duros, pero también momentos de diversión. Y por supuesto a esas nuevas amistades, que sin lugar a duda, serán para siempre.

A mis verdaderos amigos y amigas, ya sean del pueblo o no…y a todas las demás personas que alguna vez se han preocupado por mí.

Y por supuesto, a los que no están, porque se han ido o porque están por llegar.

Gracias.

ÍNDICE GENERAL

ÍNDICE DE TABLAS, GRÁFICOS E ILUSTRACIONES

1. INTRODUCCIÓN

2. CONTEXTUALIZACIÓN

3. PRIMERAS APLICACIONES

4. PROGRAMACIÓN DE LA PRODUCCIÓN DE NEUMÁTICOS

RESUMEN

Este trabajo fin de grado se centra en el desarrollo y aplicación del algoritmo de generación de columnas, el cual es ampliamente utilizado en problemas de gran escala, para diferentes problemas logísticos. Nos centraremos en la adaptación de la generación de columnas del problema de asignación, el problema del transporte, y una aplicación de mayor complejidad que se centra en el proceso de optimización de fabricación de neumáticos de una gran multinacional.

No solo se verá la adaptación del algoritmo a los diferentes problemas, sino que también se llevará a cabo su implementación, mediante el uso de un lenguaje de programación matemática, llamado Mosel utilizando para ello el programa Xpress IVE. Además, se recogerán, se analizarán y se comentarán los resultados obtenidos.

Palabras claves:

Método de generación de columnas, problema de asignación, problema del transporte, optimización de un proceso productivo, programación lineal de gran escala.

Abstract

This degree final project is focused on the development and implementation of the column generation algorithm which is widely used in large-scale problems, for different logistical problems. We will concentrate on the adaptation of the column generation for the assignment problem, the transport problem and the optimization on the process of tire manufacturing in a large multinational company. This latter application will be more complete and complex than the two previous ones.

We will not only adapt the algorithm to different problems, but also we will carry out its implementation, because of using a mathematical program language called Mosel by using the Xpress IVE program. Furthermore, the data obtained will be collected, analyzed and discussed.

Key words:

Column generation method, assignment problem, transport problem, optimization of a production process, large-scale linear programming.

Capítulo 1. Introducción

1.1. Alcance

El objetivo del presente trabajo fin de grado es el de centrarnos en una parte de la programación lineal como es el método de generación de columnas. Se pretende llevar a cabo un acercamiento a dicho tema con el fin de hacer comprender el motivo de su uso y sus aplicaciones, siendo este el principal objetivo.

Para ello, haremos un repaso desde sus orígenes, junto con algunas aplicaciones, analizando los datos correspondientes para una mejor compresión.

1.2. Motivación personal

Desde el inicio de mis estudios universitarios, siempre me he sentido atraído por la planificación de tareas, aunque para encontrar el verdadero motivo de la realización de este trabajo fin de grado, nos tenemos que remontar al tercer curso de mis estudios, cuando cursé la asignatura de Métodos Cuantitativos en Ingeniería de Organización. Durante el transcurso de dicha asignatura me fui sintiendo más absorbido por el tema de la planificación de la producción y los distintos problemas de distribución en flujo de redes. Por ello, decidí realizar este TFG, con el objetivo de continuar un poco más con mi formación en estos temas tan ampliamente tratados en la vida laboral.

1.3. Estado del arte

El método o algoritmo de generación de columnas ha sido bastante estudiado desde la pasada década de los años 50, cuando en 1958 Ford y Fulkenson probaron a resolver un problema de flujo de redes no utilizando implícitamente todas las variables. Pero no fue hasta 1960 cuando Dantzig y Wolfe consiguieron obtener una solución a partir de una estrategia desarrollada para extender las columnas de un problema lineal. Gilmore y Gomory implementaron esta técnica en 1961 para resolver el famoso problema del cutting stock o problema de corte, probablemente una de las principales aplicaciones de dicha técnica.

Desde entonces, y debido a su apropiada aplicación para problemas de gran escala que no tienen un gran número de restricciones, han sido numerosos

los estudios e investigaciones realizados al respecto. Sus aplicaciones varían desde problemas tan sencillos como puede ser al problema de asignación o el problema del transporte (cuyo modelo completo funciona correctamente), hasta tan complejos como el de ruteo de vehículos, flujos de coste mínimo o flujo de redes, o incluso, el de optimización del proceso productivo de neumáticos.

Centrándonos en la optimización del proceso productivo de neumáticos, nos hemos basado en un gran artículo sobre la esquematización del sistema productivo de neumáticos para Bridgestone / Firestone; "*A Tire Scheduling System for Bridgestone/Firestone Off-The-Road*".

Dicho modelo fue elaborado por Zeger Degraeve en Julio de 1996, aunque no fue aprobado hasta Junio 1997 por Linus Schrage en la universidad de Chicago. Tras su implementación en el sistema de fabricación de Bridgestone / Firestone, durante el primer día de funcionamiento, se consiguió producir hasta 3 neumáticos más de los que se producía mediante el sistema anterior que se tenía implantado en dicha compañía. Hasta 4 neumáticos más fueron fabricados en el segundo día de funcionamiento. Además, todo esto alejaba a la compañía de los grandes quebraderos de cabeza para intentar, manual y mentalmente, optimizar los calentadores.

Posteriormente, se han desarrollado diferentes interfaces de Windows para dicho programa, las cuales se encargan de generar todos los datos automáticamente y dar la orden de fabricación. Además, tras la ejecución, se mostraban gráficas con el apilamiento de los neumáticos y utilización de los calentadores.

Dicho artículo fue publicado en la edición de *Operations Research* en Noviembre – Diciembre de 1997, recibiendo el premio por parte de la *Association of European Operational Research Societies*, como el mejor artículo del año.

Dicho premio fue recibido debido a la demostración de su verdadera aplicación a la vida real de una empresa. También se reconoció su originalidad así como el avanzado uso de técnicas de investigación y fundamentos matemáticos utilizados. El premio fue entregado a los autores en la *Joint International EURO XV/INFORMS XXXIV Meeting*, la cual tuvo lugar en Barcelona.

1.4. Estructura del trabajo fin de grado

Inicialmente se ha incluido un resumen, tanto en español como en inglés, junto con 5 palabras claves. Posteriormente se presentan dos índices, el primero de ellos general, mientras que el segundo se trata de un índice referente a tablas, gráficos e ilustraciones.

En este primer capítulo se ha llevado a cabo una introducción referente al objetivo, la motivación y un sencillo estudio del arte sobre el tema

En el siguiente capítulo, se llevará a cabo una centralización de la técnica de generación de columnas. Se revisarán los elementos básicos de un problema de programación lineal, cómo se lleva a cabo la resolución de un problema tipo de programación lineal así como la resolución particular mediante el método simplex, y la explicación de la técnica de generación de columnas.

En el capítulo 3 se llevará a cabo la aplicación del algoritmo de generación de columnas al problema del transporte y al problema de asignación. Para ambos, se verá un planteamiento del problema junto con sus posibles variantes, su adaptación a la generación de columnas, la programación mediante un lenguaje de programación y el análisis de resultados.

Continuando con el capítulo 4 se llevará a cabo una nueva aplicación basada en la optimización de un proceso productivo de neumáticos. Para ello, se llevará a cabo una conceptualización y explicación del problema, así como la programación y el análisis de resultados correspondientes.

Tras el capítulo 4, se añaden una serie de conclusiones, impresiones y aclaraciones sobre el presente trabajo fin de grado.

Y finalmente, se detalla la principal bibliografía tratada para la elaboración de dicho trabajo.

Capítulo 2. Contextualización

2.1. Introducción a la programación lineal

2.1.1. Definición del problema de programación lineal

Un problema de programación lineal es un problema de optimización cuya función objetivo es lineal, pudiendo ser maximizada o minimizada, y sujeta a unas restricciones lineales, pudiendo ser estas tanto de igualdad como de desigualdad. Podemos suponer un problema de Programación lineal como sigue,

$$\text{Minimizar} \quad z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

$$\text{Sujeto a,} \quad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2$$

$$\ldots \quad \ldots \quad \ldots \quad \ldots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m$$

$$x_1, x_2, \ldots, x_n \geq 0$$

Dicho problema, puede ser expuesto en forma estándar, en el que solo existen restricciones de igualdad, salvo la condición de no negatividad,

$$\text{Minimizar} \quad z = cx$$

$$\text{Sujeto a,} \quad \mathrm{Ax} = \mathrm{b}, \quad \mathrm{x} \geq 0$$

Donde z es la función objetivo, la cual está formada por los coeficientes de costos conocidos c_1, c_2, … , c_n y por las variables de decisión x_1, x_2, …, x_n que deben determinarse . En cuanto a las restricciones, por un lado tenemos las restricciones de no negatividad x_1, x_2, … , $x_n \geq 0$ para cada una de las variables. Y por otro lado tenemos la igualdad $\mathrm{Ax} = \mathrm{b}$, donde A es la matriz de restricciones formada por los diferentes coeficientes tecnológicos a_{ij}, y b es el vector columna del lado derecho formado por los diferentes coeficientes b_j, que representa los requerimientos que deben satisfacerse. Dicho problema está dimensionado por n, el número de variables (representadas mediante el índice j) y m, el número de restricciones (representadas mediante el índice i).

Todo problema de programación lineal debe de cumplir las siguientes suposiciones

A) Proporcionalidad. Partiendo de una variable x_j, su aportación a la función objetivo será $c_j x_j$, así como $a_{ij} x_j$ a la i-ésima restricción.

B) Aditividad. Mediante esta propiedad se intenta garantizar, por ejemplo, que el valor total de la función objetivo se deba a la suma de cada uno de los costos individualmente.

C) Divisibilidad. Permite que las variables de decisión puedan tomar valores no enteros.

2.1.2. Conceptos básicos de programación lineal

Solución factible: Se define una solución factible como aquella que esté formada por valores de la variable x que cumple con todas las restricciones.

Solución factible óptima: Se define como la solución factible que ofrece el óptimo de la función objetivo, pudiendo existir varias, aunque no es lo habitual.

Matriz básica: Es una matriz B cuadrada de tamaño m x m extraída de las columnas de la matriz de restricciones A, que debe ser no singular, es decir con determinante distinto de cero, con el fin de que la solución sea única. Por consiguiente, las m variables x asociadas a las columnas de B son renombradas como variables básicas.

Matriz no básica: Es la matriz N originada por las columnas de A que no se encuentran en B. Del mismo modo, se denomina variables no básicas a las n – m restantes variables de x no denominadas básicas.

Dicho de otro modo, podemos explicar la matriz básica y no básica, como una descomposición de la matriz de restricciones, es decir, A = [B N]. Al igual que las variables básicas y no básicas llevando a cabo una descomposición del vector X inicial, $X = [X_B\ X_N]^T$.

Solución básica: Se dice que una solución del problema de programación lineal es básica cuando las variables no básicas toman valor cero, independientemente del valor tomado por las variables básicas. Expresado de forma matricial, cuando

$$\left.\begin{array}{l} [B\ N]\begin{bmatrix} X_B \\ X_N \end{bmatrix} = b \\ X_N = 0 \end{array}\right\} \rightarrow X_B = B^{-1}b, \text{ que es la solución básica del problema asociada a B}$$

Solución básica factible: Se trata de una solución básica en la que además las variables básicas tienen que cumplir con el principio de no negatividad, es decir, que dichas variables básicas sean factibles, $X_B \geq 0$.

Todas las soluciones que satisfagan las restricciones del problema, forman un poliedro en el espacio conocido como la región factible, donde se van a encontrar todas las soluciones factibles además de la solución óptima. Cada vértice de ese poliedro va a llevar asociado una solución factible básica.

En general, y a modo de curiosidad, puede determinarse el número máximo de soluciones básicas factibles a partir de

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

2.1.3. Ejemplo de aplicación

A continuación veremos un pequeño ejemplo para ver con más claridad los conceptos explicados.

Supongamos un ejemplo con las siguientes restricciones,

$$x_2 \leq 5$$

$$x_1 + x_2 \leq 7$$

$$x_1, x_2 \geq 0$$

Que expuesto de manera estándar con sus correspondientes holguras quedaría como sigue,

$$x_2 + x_3 = 5$$

$$x_1 + x_2 + x_4 = 7$$

$$x_1, x_2, x_3, x_4 \geq 0$$

De donde podemos extraer la matriz A = $[a_{ij}]$, de restricciones,

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Recordando la definición de matriz básica B, se trata de una matriz cuadrada m x m extraída de A, siendo la matriz no básica N, la matriz formada por el resto de columnas de A que no se encuentren en B. Podemos determinar que

en nuestro caso, la matriz básica será una matriz cuadrada de tamaño 2 x 2, presentándose a continuación las diferentes posibilidades;

1) $B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

2) $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

3) $B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

4) $B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$

5) $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Como podemos observar, tenemos 5 matrices básicas correspondiente a cinco soluciones básicas, más una base que no pudo ser formada debido a la repetición de dos columnas (columnas 1 y 4), lo que daría lugar a las seis máximas posibles soluciones según la formula,

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!} = \frac{4!}{2!\,(4-2)!} = 6$$

Ahora vamos a comprobar si cada una de estas 5 posibles soluciones básicas, son factibles o no. Para ello, tenemos que recordar que una solución básica factible, es una solución básica en la que además, las variables básicas no son negativas.

1) $B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$

$$X_B = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = B^{-1}b = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

$$X_N = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Debido a que se cumple que $X_B \geq 0$, nos encontramos ante una solución básica factible.

2) $B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$$X_B = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = B^{-1}b = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$$

$$X_N = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Debido a que se cumple que $X_B \geq 0$, nos encontramos ante una solución básica factible.

3) $B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$

$$X_B = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = B^{-1}b = \begin{bmatrix} 0 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 7 \\ -2 \end{bmatrix}$$

$$X_N = \begin{bmatrix} x_1 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Debido a que no se cumple que $X_B \geq 0$, nos encontramos ante una solución básica no factible.

4) $B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$

$$X_B = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} = B^{-1}b = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$X_N = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Debido a que se cumple que $X_B \geq 0$, nos encontramos ante una solución básica factible.

5) $B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$X_B = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = B^{-1}b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

$$X_N = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Debido a que se cumple que $X_B \geq 0$, nos encontramos ante una solución básica factible.

Como hemos podido comprobar, de las 5 soluciones básicas, solo 4 son además factibles: $X_1 = (2, 5, 0, 0)^T$; $X_2 = (7, 0, 5, 0)^T$; $X_3 = (0, 5, 0, 2)^T$; $X_4 = (0, 0, 5, 7)^T$.

En el espacio bidimensional (x_1, x_2), estas soluciones corresponden a X_1= (2, 5); X_2= (7, 0); X_3= (0, 5); X_4= (0, 0), las cuales forman una región en el espacio, la denominada región factible.

A continuación se muestra un gráfico bidimensional donde se ha llevado a cabo la representación de las soluciones anteriores, pudiéndose observar la región factible formada por dichas soluciones.

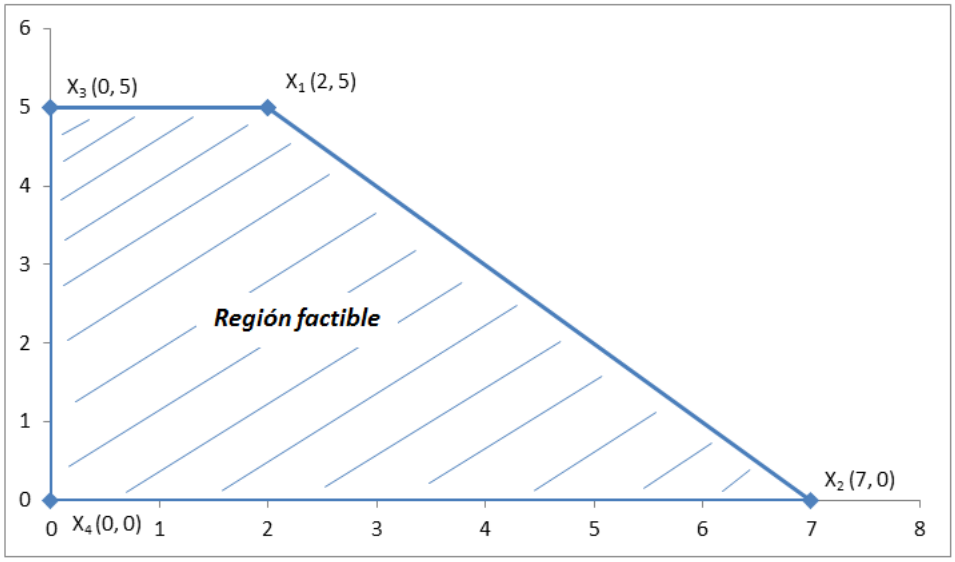

Ilustración 1

A continuación se resumen los dos teoremas fundamentales de la teoría de la programación lineal, cuya teoría más desarrollada puede observarse en el libro de Mokhtar S. Bazaraa, John J. Jarvis.

> Teorema 1. Mediante el cual se nos asegura que cada vértice corresponde con una solución básica factible.
>
> Teorema 2. Este teorema nos asegura que si existe una solución óptima en un problema de programación lineal, entonces existe un vértice que es óptimo. De forma equivalente, también existirá una solución básica factible óptima.

2.2. Método simplex

Mediante el teorema fundamental de la programación lineal, se podría llegar a la solución óptima tras la evaluación de la función objetivo en un número finito de vértices o soluciones básicas factibles. Sin embargo, existen problemas de gran complejidad que pueden contar con un número excesivamente elevado de soluciones factibles básicas, y por ello se requiere alguna técnica para llegar más rápidamente a la solución óptima. Esta técnica es la que se conoce como el método simplex.

Los problemas de programación lineal pueden ser resueltos mediante el método simplex, desarrollado por el norte americano George Dantzig en el año 1947. Para llevar a cabo la aplicación del método simplex, debemos considerar un problema de programación lineal con formato estándar, donde las restricciones estén representadas mediante igualdades.

El método simplex parte de un vértice o solución factible inicial cualquiera, y operando de forma iterativa va pasando de un vértice a otro adyacente que mejora, o al menos no empeora la función objetivo, con el fin de llegar a la solución factible óptima en un número finito de pasos.

La siguiente ilustración muestra el recorrido del algoritmo simplex a través de un politopo, viajando de vértice en vértice hasta llegar a la solución óptima.

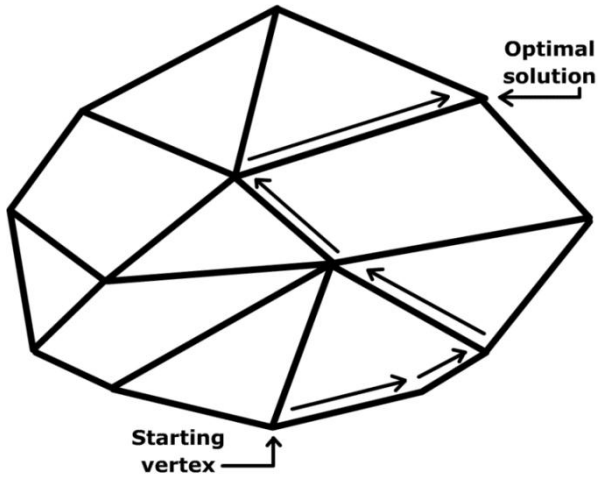

Ilustración 2

Durante la resolución de un problema de programación lineal, en cada iteración del método simplex se genera, una solución básica factible primal x' y una solución complementaria ω para el dual, conocida como vector precios sombra o multiplicadores del simplex. De igual modo pasa cuando se alcanza la solución óptima.

2.3. Generación de columnas

Hoy en día pueden resolverse problemas de gran envergadura en un ordenador normal y corriente, pero existen muchos problemas prácticos de muy grandes dimensiones que no pueden ser formulados adecuadamente debido al gran número de variables y a la cantidad de memoria exigida para almacenarlas. Estos problemas de gran escala pueden ser resueltos mediante técnicas de descomposición que transforman la solución de un gran

problema en una serie de problemas de menor complejidad. Uno de estos métodos, está basado en técnicas de generación de columnas, que toma como base el método simplex.

La técnica de generación de columnas lleva a cabo una analogía entre los conceptos de variables del problema y columnas, siendo muy útil cuando nos encontramos ante un problema que tiene un número de variables mucho mayor que el número de restricciones ($n \gg m$). La técnica del método de generación de columnas va a llevar a cabo la resolución del problema donde no todas las columnas o variables van a ser conocidas, bien debido a que estas no se conocen o debido a que no es de utilidad conocerlas debido al gran número de variables. Por ejemplo, cada columna puede corresponder a un vértice de un poliedro, o a un camino entre dos nodos de un determinado grafo… En cada caso, el conjunto de columnas tendrá un aspecto determinado, y por ello, podemos decir que el conjunto de todas las columnas es $S \subset R^m$, y que para cada posible variable existirá una columna $a \in S$ con su costo asociado.

El problema inicial expuesto en la forma de columnas se conoce como el Problema Maestro (PM),

$$\text{(PM) Minimizar} \quad z = \sum_{j \in N} c_j x_j$$

$$\text{Sujeto a,} \quad \sum_{j \in N} a_j x_j = \mathrm{b}$$

$$\mathrm{x}_j \geq 0, \quad \forall \mathrm{j} \in \mathrm{N}$$

Siendo $N = \{1, \dots, n\}$ el conjunto de índices correspondiente a cada una de las variables, con un índice $\mathrm{j} \in \mathrm{N}$ asociado a cada variable x_j y a cada columna a_j. Los vectores columna a_j y b pertenecen a R^m.

Como podremos observar posteriormente, en el Problema Maestro el número de variables n puede ser muy elevado, pudiendo llegar a ser de miles o millones de columnas, imposibles de almacenar en una memoria.

Por este motivo, se crea el Problema Maestro Restringido (PMR), en el cual únicamente se generan aquellas columnas que vamos a necesitar. De manera general, podemos suponer que se han generado un conjunto de columnas con índices $G = \{1, \dots, \tilde{\mathrm{n}}\}$, tal que $G \subset N$ y $\tilde{\mathrm{n}} \ll n$.

$$\text{(PMR) Minimizar} \quad z = \sum_{j \in G} c_j x_j$$

$$\text{Sujeto a,} \quad \sum_{j \in G} a_j x_j = \mathrm{b}$$

$$\mathrm{x}_j \geq 0, \quad \forall \mathrm{j} \in \mathrm{G}$$

Todo problema maestro restringido inicial tiene que cumplir la condición de que sea factible. De ser así, este problema puede resolverse perfectamente aplicando el método simplex.

Tras la aplicación del método simplex, se generará una solución primal x' y una solución dual $\omega = c_B B^{-1}$. Para determinar si esa solución es óptima, sería necesario calcular el coste reducido para toda columna no básica a_j para después tomar el coste reducido máximo. El coste reducido de una variable con columna asociada a_j se puede calcular mediante la fórmula (ver Bazaraa-Jarvis capítulo 3 o también el artículo de Programación a gran escala de Andrés L. Medaglia.):

$$z_j - c_j = c_B B^{-1} a_j - c_j = \omega\, a_j - c_j$$

Es un resultado bien conocido en programación lineal que si $z_j - c_j \leq 0$ para todo j, entonces la solución básica actual es una solución óptima.

Por este motivo, vamos a calcular el coste reducido máximo $Máx_{j=1,\dots ñ}\{z_j - c_j\}$ dando lugar al siguiente subproblema P_ω,

$$(P_\omega) \quad \underset{a \in S}{Máx} \{\omega a - c(a)\}$$

Tras la solución de dicho subproblema, habremos generado una columna a' tal que $a' \in S$ y que sea una solución óptima del subproblema. Si $\omega a' - c(a') \leq 0$, entonces habremos encontrado la solución óptima del problema. Por el contrario, si $\omega a' - c(a') > 0$, añadiremos la columna generada a' al conjunto de columnas G del Problema Maestro Restringido, y realizaremos una nueva iteración.

Siempre que una variable no básica entra en la base, nos tenemos que asegurar de que el nuevo conjunto de variables, es decir, el conjunto básico actual menos la variable de salida más la variable de entrada debe satisfacer que sea una base, que siga siendo factible y que el valor de la función objetivo, al menos, no crezca. La eliminación de una columna de la base, es llevada a cabo mediante el propio programa matemático.

Por lo tanto, se podría resumir el procedimiento de generación de columnas para resolver el Problema Maestro en 5 pasos, tal y como se muestra a continuación:

1. Se selecciona un conjunto inicial de columnas con índices G. con la única condición de que el Problema Maestro Restringido inicial sea factible.

2. Se resuelve explícitamente el Problema Maestro Restringido con las columnas actuales, cuyas soluciones son la solución primal x y la solución dual ω.

3. Usando el vector de precios sombra, se resuelve el subproblema, con solución óptima $\bar{a}$.

4. En el caso en el que $\omega\bar{a} - c(\bar{a}) > 0$, se añade la columna $\bar{a}$ generada al Problema Maestro Restringido, y se vuelve a la etapa 2.

5. En cambio, si $\omega\bar{a} - c(\bar{a}) \leq 0$, ya tenemos la solución óptima del Problema Maestro.

Como hemos podido comprobar, el algoritmo de generación de columnas está basado en la interacción entre el problema de programación lineal restringido y el subproblema encargado de generar las nuevas columnas.

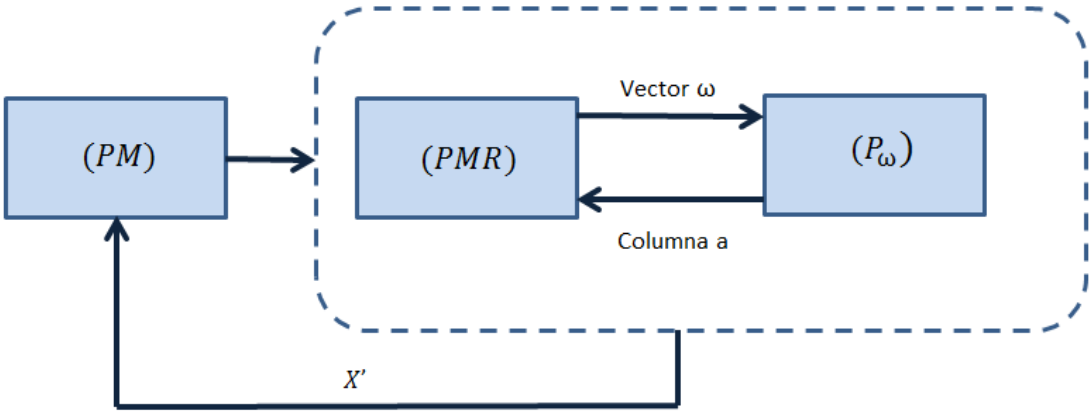

Ilustración 3

Mediante la generación de columnas, solo se generan explícitamente un número muy pequeño de todas las columnas del espacio S. Dependiendo de la estructura de S y de la forma del costo $c(a)$, el subproblema puede tratarse de un problema de programación lineal sencillo, de un problema de programación dinámica, de un problema de redes, o de un problema tipo mochila…

Entre las numerosas aplicaciones llevadas a cabo por el algoritmo de generación de columnas, podemos destacar su aplicación en el problema de Cutting stock, donde partiendo de un rollo de papel, se intenta cortar en trozos de diferentes tamaños con la idea de satisfacer a unos clientes y despilfarrar el menor papel posible. También podemos mencionar su aplicación al problema de asignación y al problema del transporte, cuya aplicación será vista en el próximo capítulo. Flujo en redes multiproducto o de tráfico, flujo de costo mínimo de una red, problemas de tripulaciones… podrían ser algunas otras de las varias aplicaciones de este algoritmo.

Algunas de estas aplicaciones, junto con otras varias son recogidas en el libro de Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, titulado Column Generation. Además, en internet y otros artículos de investigación matemática, se pueden encontrar numerosas aplicaciones de la generación de columnas.

Capítulo 3. Primeras aplicaciones

3.1. Problema del transporte

3.1.1. Introducción general al problema del transporte

El problema general del transporte hace referencia a la distribución de cualquier bien, producto o material desde cualquier grupo de centros de suministro, llamados de ahora en adelante orígenes, a cualquier grupo de centros de recepción, a los que haremos referencia como destinos. El objetivo, es el de minimizar los costos totales de la distribución sujeto a una serie de restricciones. Cada origen tiene cierto suministro de unidades que distribuir a los destinos, al igual que cada destino tiene una cierta demanda de unidades que deben recibirse de los orígenes. Podemos plantear de manera general el problema del transporte como sigue,

$$Minimizar \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$Sujeto\ a \quad \sum_{j=1}^{n} x_{ij} \leq s_i;\ \forall i = 1, \dots, m$$

$$\sum_{i=1}^{m} x_{ij} \geq d_j;\ \forall j = 1, \dots, n$$

$$x_{ij} \geq 0;\quad \forall\, i, j$$

La función a minimizar será función de x_{ij}, es decir, del número de unidades transportadas desde el origen i al destino j, y del coste c_{ij} en el que se incurre si una unidad es transportada del origen i al destino j. En cuanto a las restricciones, asociado a cada origen i existe un coeficiente s_i, que representa cantidad máxima que cada origen puede suministrar, y nunca pudiendo ser la cantidad suministrada mayor que la disponible. Esta primera restricción representa la oferta de los distintos orígenes. De igual forma, cada destino j lleva asociado un coeficiente d_j, que representa la cantidad requerida por dicho destino. Esta segunda restricción representa la demanda de los diversos destinos. Por último, la última restricción hace referencia a la no negatividad de las variables, condición más que necesaria en un problema de minimización como el que nos ocupa.

A modo curiosidad, el problema del transporte es nombrado así debido a que muchas de sus aplicaciones consisten en determinar la manera óptima de transportar productos entre diferentes lugares. No obstante, hay aplicaciones como por ejemplo, la programación de la producción, en las cuales se aplica el problema del transporte sin tener lugar un transporte como tal.

Los parámetros del modelo del problema del transporte son las ofertas s_i, las demandas d_j y los costes c_{ij}.

3.1.2. Variedades del problema del transporte.

En el modelo anterior sobre el problema del transporte, podemos definir la oferta total O_T como la suma de las ofertas individuales de los diferentes orígenes existentes $O_T = \sum_{i=1}^{m} s_i, \forall i = 1, \dots, m$. También la demanda total D_T, como la suma de las demandas individuales de los diferentes destinos $D_T = \sum_{i=1}^{n} d_i, \forall j = 1, \dots, n$.

El problema se dice que está balanceado cuando $O_T = D_T$, es decir, cuando existe un origen para cada destino. En este caso, las restricciones anteriores son cumplidas en forma de igualdad, salvo las de no negatividad, claro está.

$$Minimizar \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$Sujeto\ a \quad \sum_{j=1}^{n} x_{ij} = s_i; \ \forall i = 1, \dots, m$$

$$\sum_{i=1}^{n} x_{ij} = d_j; \ \forall j = 1, \dots, n$$

$$x_{ij} \geq 0; \quad \forall\, i, j$$

También nos podemos encontrar en una situación en la que la oferta total excede a la demanda total $O_T > D_T$. Ante esta situación, el problema podría balancearse fácilmente, añadiendo un nuevo destino ficticio con una demanda equivalente al excedente de la oferta. Las unidades envías hacia el destino ficticio representa las unidades ofertadas que no han sido utilizadas. Por ello, como no serían envíos reales, se les asigna un coste $c_k = 0$, para que así no influya en la función objetivo a minimizar.

$$Minimizar \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$Sujeto\ a \quad \sum_{j=1}^{n} x_{ij} \leq s_i; \ \forall i = 1, \dots, m$$

$$\sum_{i=1}^{n} x_{ij} = d_j; \ \forall j = 1, \dots, n$$

$$x_{ij} \geq 0; \quad \forall\, i, j$$

La última situación posible ante la que nos podemos encontrar sería el caso en el que el problema no es factible, debido a que la demanda total es mayor que la oferta total $D_T > O_T$. Ante esta situación, no se podrá cumplir con las restricciones del problema, careciendo de sentido su resolución.

Para llevar a cabo la resolución del problema del transporte mediante generación de columnas, no es necesario que el problema esté balanceado, pero sí, que la oferta total exceda a la demanda total $O_T > D_T$.

De ahora en adelante, nos referiremos al problema del transporte como aquel modelo adecuado para llevar a cabo su resolución mediante generación de columnas.

3.1.3. Resolución del problema del transporte mediante generación de columnas

Recordando del capítulo anterior, para llevar a cabo la resolución de un problema de programación lineal mediante generación de columnas, era necesario desarrollar un Problema Maestro Restringido (PMR). A partir del (PMR), es necesario resolver un subproblema (P_ω). A continuación se desarrollará para el problema del transporte en particular.

Podemos considerar el Problema Maestro Restringido como aquel, formado por un conjunto G de columnas tal que sea posible llevar a cabo una asignación entre M orígenes y N destinos

$$\text{(PMR) Minimizar} \quad \sum_{(i,j)\in G} c_{ij}x_{ij}$$

$$\text{Sujeto a,} \quad \sum_{\{j:(i,j)\in G} x_{ij} \leq s_i, \quad \mathrm{i} = 1, \dots, \mathrm{m}$$

$$\sum_{\{i:(i,j)\in G} x_{ij} = d_j, \quad \mathrm{j} = 1, \dots, \mathrm{n}$$

$$\mathrm{x}_{ij} \geq 0, \ \forall(\mathrm{i},\mathrm{j}) \in \mathrm{G}$$

Todo problema maestro restringido inicial tiene que cumplir la condición de que sea factible. Hoy en día, existen algunos métodos mediante los cuales pueden obtenerse buenas soluciones factibles iniciales para el problema del transporte. Para aplicar estos métodos, es necesario tener elaborada la tabla del transporte, que no es más que una tabla donde se recogen por filas los destinos y por columnas los orígenes. En cada una de las casillas formadas por dichas filas y columnas se indica la variable encargada x_{ij} así como el coste asociado c_{ij}.

Una tabla del transporte puede tener un aspecto similar a la siguiente;

	ORÍGENES			Total Ofertas
DESTINOS	$x_{11} \mid c_{11}$	…	$x_{1n} \mid c_{1n}$	s_1
	…	…	…	…
	$x_{1m} \mid c_{1m}$	…	$x_{nm} \mid c_{nm}$	s_m
Total demanda	d_1	…	d_n	

Tabla 1

Entre los métodos que hablábamos anteriormente, podemos mencionar;

a) Método de la esquina noroeste

Este método utiliza una tabla similar a la anterior y comienza por la esquina más noroeste de la tabla, esta es, la que se encuentra más cerca de la esquina superior izquierda. A la variable correspondiente se le asignará el mayor valor posible, es decir, el mínimo entre la oferta o la demanda. Pueden darse varias situaciones;

1) Si $x_{11} = s_1 = d_1$, entonces no podrá haber ninguna variable más distinta de cero ni en la primera fila $\sum_{j=2}^{n} x_{1j} = 0$, ni en la primera columna $\sum_{i=2}^{n} x_{i1} = 0$.

2) Si $x_{11} = s_1$, entonces no podrá haber ninguna variable distinta de cero en la primera fila $\sum_{j=2}^{n} x_{1j} = 0$. Además tendremos que sustituir el valor inicial de d_1 por el valor que aún queda por asignar.

3) En el caso en el que $x_{11} = d_1$, sucederá algo de manera muy similar a la situación anterior, pero afectando a columnas en vez de a filas.

Tras haber finalizado la primera iteración, el proceso se repite eligiendo siempre la celda disponible más cerca de la esquina superior izquierda.

b) Método del coste mínimo o método greedy.

El método anterior no tiene en cuenta los costos, y nos puede proporcionar una solución muy alejada de la óptima. Mediante este método, se busca la variable x_{ij} con menor coste, asignándola el máximo valor que pueda tomar, presentándose alguna de las situaciones vistas en el anterior método. Posteriormente, se realizarán

nuevas iteraciones, buscando entre las variables disponibles, las de menor costo.

c) Método de Vogel

Para cada fila / columna, se calcula una penalización mediante la diferencia entre los dos costes más pequeños. Posteriormente, se elige la fila / columna que tiene la penalización más alta, eligiéndose la variable x_{ij} con el coste más pequeño. Entonces se procede de manera análoga a los dos anteriores métodos.

En cualquier caso, sea cual sea la forma elegida para tomar la solución factible inicial, tras resolver el Problema Maestro Restringido mediante el método simplex, se generaría una solución dual asociada a dos vectores duales $\vec{\omega}^1 \in R^m, \vec{\omega}^2 \in R^n$.

Para determinar si esa solución es óptima, es necesario calcular el coste reducido de la columna (i,j) . Dado que la variable X_{ij} aparece una única vez en la fila i (con posición i) y una única vez en fila j (con posición m + j), el coste reducido se calcula como sigue,

$$z_{ij} - c_{ij} = \vec{\omega} a_{ij} - c_{ij} = \vec{\omega}_i^1 + \vec{\omega}_j^2 - c_{ij}$$

Y por lo tanto, en el subproblema habrá que calcular cual es el máximo coste reducido entre los elementos de la matriz, tal que

$$(P_\omega) \quad \underset{(i,j)}{Máx} \left\{ \vec{\omega}_i^1 + \vec{\omega}_j^2 - c_{ij} \right\} = \vec{\omega}_k^1 + \vec{\omega}_l^2 - c_{kl}$$

Si se obtuvo que $\vec{\omega}_k^1 + \vec{\omega}_l^2 - c_{kl} \leq 0$, nos encontraremos ante la solución óptima. En el caso en el que $\vec{\omega}_k^1 + \vec{\omega}_l^2 - c_{kl} > 0$, se habrá generado una nueva variable x_{kl}, diciéndose que se ha generado una nueva columna. Dicha columna, será añadida a G, volviéndose a resolver el Problema Maestro Restringido con esa nueva columna.

3.1.4. Programación del problema del transporte mediante generación de columnas

A continuación se ha llevado a cabo una programación en Mosel, mediante el programa Xpress IVE, referente al problema del transporte mediante generación de columnas. Inicialmente, se ha establecido un tamaño de 9x13, con dos criterios de parada; O bien tras encontrar la solución óptima, o bien tras un número determinado de iteraciones máximas. Como solución inicial, se ha tomado aquella calculada mediante el método Greedy.

Con el fin de poder establecer una comparación entre el número posible de columnas susceptibles de ser generadas con respecto al número de columnas generadas, se ha desarrollado un pequeño apartado dentro del mismo código fuente, que nos permitirá comparar la diferencia existente. De forma similar, se ha llevado a cabo una implementación para comparar la solución inicial del método Greedy y la solución óptima, y observar la diferencia existente entre ambas soluciones.

A continuación se adjunta el código fuente correspondiente.

```
(! Problema de transporte, con datos generados aleatoriamente,
resuelto mediante generación de columnas, habiéndose tomado
como solución inicial aquella proporcionada por el método Greedy!)

model "Problema del transporte"
uses "mmxprs"; !gain access to the Xpress-Optimizer solver

!Declaramos las variables
declarations
        m=9
        n=13
        origenes = 1..m
        destinos = 1..n
        ofe: array(origenes) of integer
        dem: array(destinos) of integer
        costo: array(origenes,destinos) of integer
        x: array(origenes,destinos) of integer
        y: array(origenes,destinos) of mpvar
        u:array(origenes,destinos)of real

        final:integer
        ofe_res:array(origenes) of integer
```

```
        dem_res: array(destinos) of integer

        res_ofe:array(origenes) of linctr
        res_dem: array(destinos) of linctr

        w1:array(origenes) of real
        w2:array(destinos) of real
        ki, kj, iter:integer
        dmax:real

        tol = 0.001
        iter_max = 100000
        base: basis
        ncol: integer
        costo_total:linctr
        minimo: integer
        salida_completa = 1

end-declarations


!Para que se repitan los mismos resultados, activamos:
setrandseed(12345)


!Generamos las ofertas y las demandas de manera aleatoria, así como los
costos
forall(i in origenes)ofe(i):=100+round(100*random)
forall(j in destinos)dem(j):=60+round(40*random)
forall(i in origenes,j in destinos)costo(i,j):=10+round(90*random)
forall(i in origenes,j in destinos)u(i,j):=minlist(ofe(i),dem(j))
oferta_total:=sum(i in origenes)ofe(i)
demanda_total:=sum(j in destinos)dem(j)

writeln("Problema de transporte, m = ",m,", n = ",n,"\n")
writeln("oferta total = ",oferta_total)
writeln("demanda total = ",demanda_total)


!Nos podemos encontrar en una situación en la que el problema no sea
factible.
if(oferta_total < demanda_total)then
```

```
  writeln("Problema no factible, OT < DT")
  exit(0)
end-if


!Si queremos imprimir la matriz de costos
(!
writeln ("La matriz de costos es la siguiente:")
writeln
forall(i in origenes)do
        writeln
        forall (j in destinos)
                write ("\t",costo(i,j))
                writeln
        end-do
writeln
writeln !)


!!!SOLUCIÓN GREEDY !!!
!Vamos a calcular la solución inicial
!Será tomada como solución inicial para el método de generación de columnas

!Heurística greedy
writeln("\n\nHeurística greedy")

forall(i in origenes)ofe_res(i):=ofe(i)
forall(j in destinos)dem_res(j):=dem(j)
final:=0
iter:=0
dem_serv:=0

while(final=0)do
 iter:=iter+1
 aux:=9999

!Buscamos el menor coste y que además, sea posible enviar y recibir pedidos
 forall(i in origenes,j in destinos|ofe_res(i)>0 and dem_res(j)>0)do
  if(costo(i,j)<aux)then
    ik:=i
    jk:=j
```

```
      aux:=costo(i,j)
    end-if
  end-do

!cuando encontramos el mínimo coste, actualizamos y guardamos datos
  x(ik,jk):=minlist(ofe_res(ik),dem_res(jk))
  ofe_res(ik):=ofe_res(ik)-x(ik,jk)
  dem_res(jk):=dem_res(jk)-x(ik,jk)
  dem_serv:=dem_serv+x(ik,jk)
!Hasta que se ha servido a toda la demanda
  if(dem_serv=demanda_total)then
     final:=1
  end-if
end-do !del do-while


!Calculamos costo e imprimimos resultados
z_greedy:=sum(i in origenes,j in destinos)costo(i,j)*x(i,j)
writeln("Heurística greedy Costo total = ",z_greedy)
writeln

forall(j in destinos)write("\tD", j)
forall(i in origenes)do
                    writeln
                    write("O",i)
          forall(j in destinos)write("\t", x(i,j))
end-do
writeln
writeln


! Problema Maestro inicial
!Creamos las variables y correspondientes
forall(i in origenes,j in destinos)
if(x(i,j)>0)then
   create (y(i,j))
   costo_total+=costo(i,j)*y(i,j)
   ncol:=ncol+1
end-if

forall(i in origenes)res_ofe(i):=
  sum(j in destinos|x(i,j)>0)y(i,j)<=ofe(i)
```

```
forall(j in destinos)res_dem(j):=
  sum(i in origenes|x(i,j)>0)y(i,j)=dem(j)

exportprob(EP_MIN,"",costo_total)

minimize(costo_total)

writeln("objetivo greedy = ",getobjval)

!!!!GENERACIÓN DE COLUMNAS !!!!
! Bucle de generación de columnas:
final:=0
iter :=0

repeat
          iter+=1
          minimize(costo_total)
          savebasis(base)


!Calculamos la solución dual
          forall(i in origenes)do
      w1(i):=res_ofe(i).dual
       !writeln("\n w1 \t", w1(i))
    end-do
   forall(j in destinos)do
     w2(j):=res_dem(j).dual
   end-do


!Buscamos la solución dual máxima
          dmax:=-99999
          forall(i in origenes,j in destinos)do
            if(w1(i)+w2(j)-costo(i,j) > dmax) then
              dmax:=w1(i)+w2(j)-costo(i,j)
              ki:=i
              kj:=j
            end-if
          end-do


!Añadimos la variable /columna con solución dual máxima
```

```
        if(dmax <= tol)then final:=1
        else  ! aquí añado la nueva columna

                if(salida_completa=1)then
                  writeln("iteracion ",iter,", costo actual =",getobjval,
                          ", se va a annadir la columna (",ki,",",kj,")")
                end-if

                ncol+=1
                create(y(ki,kj))

                !Calculamos el Nuevo costo total
                costo_total+=costo(ki,kj)*y(ki,kj)

                res_ofe(ki)+=y(ki,kj)
                res_dem(kj)+=y(ki,kj)

                loadprob(costo_total)
                loadbasis(base)
   end-if


!Establecemos un criterio de parada
until (final =1 or iter >iter_max)


!Imprimimos resultados por pantalla
writeln("\nColumnas posibles = ",m*n,", columnas generadas = ",ncol)
writeln(" Columnas generadas (%) =" ,100-100*(m*n-ncol)/(m*n) ," %")
writeln("\n\nProblema actual. Costo total = ",getobjval)

if(salida_completa=1)then
  writeln("\nSolucion optima:\n")
  forall(j in destinos)write ("\tD ",j)
 forall(i in origenes)do
        writeln
   write("O",i)
   forall(j in destinos)write("\t",y(i,j).sol)
 end-do

  writeln("\n\n")
```

```
  exportprob(0,"",costo_total)


!diferencia entre solución inicial y solución óptima en %.
hueco:=100*(z_greedy-getobjval)/getobjval
writeln("Hueco (%) = ",hueco, " %")

end-if
end-model
```

3.1.5. Resultados experimentales. Problema del transporte

En este apartado nos centraremos en ir explicando la salida del programa Mosel tras la ejecución del código fuente anterior.

Para una matriz de costos generada aleatoriamente (la cual, no ha sido impresa por pantalla), y teniendo en cuenta las restricciones de la oferta y de la demanda total (también generadas aleatoriamente), se calcula una solución Greedy inicial, con un costo asociado. Se genera así la primera matriz que nos indica las cantidades transportadas entre los orígenes (por filas) y los destinos (por columnas). Todo ello, determinado mediante una función objetivo a minimizar, sujeto a una serie de restricciones, tal y como sigue.

Problema de transporte, m = 9, n = 13

oferta total = 1412
demanda total = 1078

Heurística greedy
Heurística greedy Costo total = 25665

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13
01	0	0	0	0	0	36	0	0	0	97	0	0	0
02	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	0	0	0	34	0	0	81	0	42	0	0
04	37	0	0	0	0	0	99	0	0	0	42	0	0
05	0	0	0	0	0	0	0	81	0	0	0	95	0
06	0	0	0	74	50	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	0	0	0	0	0	0	71
08	60	0	61	0	0	0	0	0	0	0	0	0	0
09	0	88	0	0	30	0	0	0	0	0	0	0	0

Tabla 2

\ Using Xpress-MP extensions
Minimize

19 y(1,6) + 15 y(1,10) + 28 y(3,6) + 19 y(3,9) + 38 y(3,11) + 21 y(4,1) + 12 y(4,7) + 36 y(4,11) + 17 y(5,8) + 21 y(5,12) + 79 y(6,4) + 23 y(6,5) + 30 y(7,13) + 17 y(8,1) + 12 y(8,3) + 14 y(9,2) + 16 y(9,5)

Subject To
res_dem(13): y(7,13) = 71
res_dem(12): y(5,12) = 95
res_dem(11): y(3,11) + y(4,11) = 84
res_dem(10): y(1,10) = 97
res_dem(9): y(3,9) = 81
res_dem(8): y(5,8) = 81
res_dem(7): y(4,7) = 99
res_dem(6): y(1,6) + y(3,6) = 70
res_dem(5): y(6,5) + y(9,5) = 80
res_dem(4): y(6,4) = 74
res_dem(3): y(8,3) = 61
res_dem(2): y(9,2) = 88
res_dem(1): y(4,1) + y(8,1) = 97
res_ofe(9): y(9,2) + y(9,5) <= 118
res_ofe(8): y(8,1) + y(8,3) <= 121
res_ofe(7): y(7,13) <= 158
res_ofe(6): y(6,4) + y(6,5) <= 191
res_ofe(5): y(5,8) + y(5,12) <= 198
res_ofe(4): y(4,1) + y(4,7) + y(4,11) <= 178
res_ofe(3): y(3,6) + y(3,9) + y(3,11) <= 195
res_ofe(2): <= 120
res_ofe(1): y(1,6) + y(1,10) <= 133

Bounds
End

Posteriormente, a partir de esa solución inicial, se entrará en un bucle de generación de columnas en las que se irán incorporando nuevas columnas que nos permitirán alcanzar la solución óptima. Y es aquí, donde podremos observar una pequeña comparativa entre el número de columnas generadas (tan solo 22) con respecto al número de columnas posibles (117). Para un problema bastante pequeño, hemos llegado a la solución óptima utilizando tan solo un 18.8034 % de todas las columnas, dándonos una idea de la gran utilidad del algoritmo de generación de columnas para grandes problemas.

Tras esto, se ha generado una nueva matriz con las cantidades óptimas a transportar entre orígenes y destinos para satisfacer así, todas las restricciones.

objetivo greedy = 25665
iteracion 1, costo actual =25665, se va a annadir la columna (4,4)
iteracion 2, costo actual =24867, se va a annadir la columna (5,7)
iteracion 3, costo actual =24405, se va a annadir la columna (9,3)
iteracion 4, costo actual =24139, se va a annadir la columna (8,11)
iteracion 5, costo actual =23971, se va a annadir la columna (2,12)

Columnas posibles = 117, columnas generadas = 22
Columnas generadas (%) =18.8034 %

Problema actual. Costo total = 23587
Solucion optima:

	D 1	D 2	D 3	D 4	D 5	D 6	D 7	D 8	D 9	D 10	D 11	D12	D13
O1	0	0	0	0	0	36	0	0	0	97	0	0	0
O2	0	0	0	0	0	0	0	0	0	0	0	64	0
O3	0	0	0	0	0	34	0	0	81	0	0	0	0
O4	91	0	0	74	0	0	13	0	0	0	0	0	0
O5	0	0	0	0	0	0	86	81	0	0	0	31	0
O6	0	0	0	0	80	0	0	0	0	0	0	0	0
O7	0	0	0	0	0	0	0	0	0	0	0	0	71
O8	6	0	31	0	0	0	0	0	0	0	84	0	0
O9	0	88	30	0	0	0	0	0	0	0	0	0	0

Tabla 3

\ Using Xpress-MP extensions
Minimize

19 y(1,6) + 15 y(1,10) + 25 y(2,12) + 28 y(3,6) + 19 y(3,9) + 38 y(3,11) + 21 y(4,1) + 56 y(4,4) + 12 y(4,7) + 36 y(4,11) + 14 y(5,7) + 17 y(5,8) + 21 y(5,12) + 79 y(6,4) + 23 y(6,5) + 30 y(7,13) + 17 y(8,1) + 12 y(8,3) + 22 y(8,11) + 14 y(9,2) + 13 y(9,3) + 16 y(9,5)

Subject To
res_dem(13): y(7,13) = 71
res_dem(12): y(2,12) + y(5,12) = 95
res_dem(11): y(3,11) + y(4,11) + y(8,11) = 84
res_dem(10): y(1,10) = 97
res_dem(9): y(3,9) = 81
res_dem(8): y(5,8) = 81
res_dem(7): y(4,7) + y(5,7) = 99
res_dem(6): y(1,6) + y(3,6) = 70

res_dem(5): y(6,5) + y(9,5) = 80
res_dem(4): y(4,4) + y(6,4) = 74
res_dem(3): y(8,3) + y(9,3) = 61
res_dem(2): y(9,2) = 88
res_dem(1): y(4,1) + y(8,1) = 97
res_ofe(9): y(9,2) + y(9,3) + y(9,5) <= 118
res_ofe(8): y(8,1) + y(8,3) + y(8,11) <= 121
res_ofe(7): y(7,13) <= 158
res_ofe(6): y(6,4) + y(6,5) <= 191
res_ofe(5): y(5,7) + y(5,8) + y(5,12) <= 198
res_ofe(4): y(4,1) + y(4,4) + y(4,7) + y(4,11) <= 178
res_ofe(3): y(3,6) + y(3,9) + y(3,11) <= 195
res_ofe(2): y(2,12) <= 120
res_ofe(1): y(1,6) + y(1,10) <= 133

Bounds

End

Hueco (%) = 8.80994 %

Finalmente, se muestra por pantalla el valor de una variable denominada Hueco, que expresa en % la diferencia entre la solución inicial y la solución óptima, pudiéndonos hacer una idea de la buena aproximación de la solución inicial.

A continuación, se muestra una tabla resumen en la que se recogen los datos correspondientes a los resultados de varias simulaciones para problemas de mayores tamaños.

m	n	Col. posibles	Columnas generadas	Columnas generadas (%)	Solución inicial	Solución óptima	Hueco (%)
9	13	117	22	18.80 %	25.665	23.587	8.8094 %
20	26	**520**	**49**	**9.42 %**	**42.786**	**38.759**	**10.39 %**
42	50	2.100	129	6.14 %	89.198	78.407	13.76%
70	77	**5.390**	**209**	**3.88 %**	**164.018**	**146.200**	**12.19 %**
100	110	11.000	291	2.65 %	237.345	**220.114**	7.83 %
241	250	**60.250**	**727**	**1.21%**	**550.516**	**516.854**	**6.51 %**
498	506	251.988	1.550	0.62 %	1.320.660	1.258.130	4.97 %
985	1.000	**985.000**	**2.725**	**0.277%**	**3.171.126**	**3.113.830**	**1.84%**
2.500	2.515	6.287.500	5.917	0.094 %	14.766.455	14.637.200	0.88%
4.000	4.022	**16.088.000**	**8.988**	**0.0559%**	**44.807.071**	**44.630.300**	**0,396%**

Tabla 4

Como podemos observar, a medida que incrementamos el tamaño de m y n, consecuentemente se va incrementando tanto el número de columnas posibles para ser generadas como el número de columnas generadas. Sin embargo, podemos apreciar como el porcentaje de columnas generadas va decrementando a medida de se aumenta el tamaño del problema. Para ello, debemos fijarnos en la 5 columna, columnas generadas (%).

Y es aquí donde podemos encontrar la principal utilidad de la técnica de generación de columnas. Para problemas inmensamente grandes, del orden de 16.088.000 variables, tan solo hemos necesitado utilizar 8.988 para llegar a la solución óptima. Todo esto se traduce en una utilización de variables de 0.0559%

A continuación se muestran una gráfica donde se muestra de forma más visual los resultados recogidos en la tabla anterior. Para el 100% de columnas posibles, vemos la proporción de columnas que sí que han sido generadas y la proporción de columnas que no, pudiendo observar además como se va decrementando ese porcentaje.

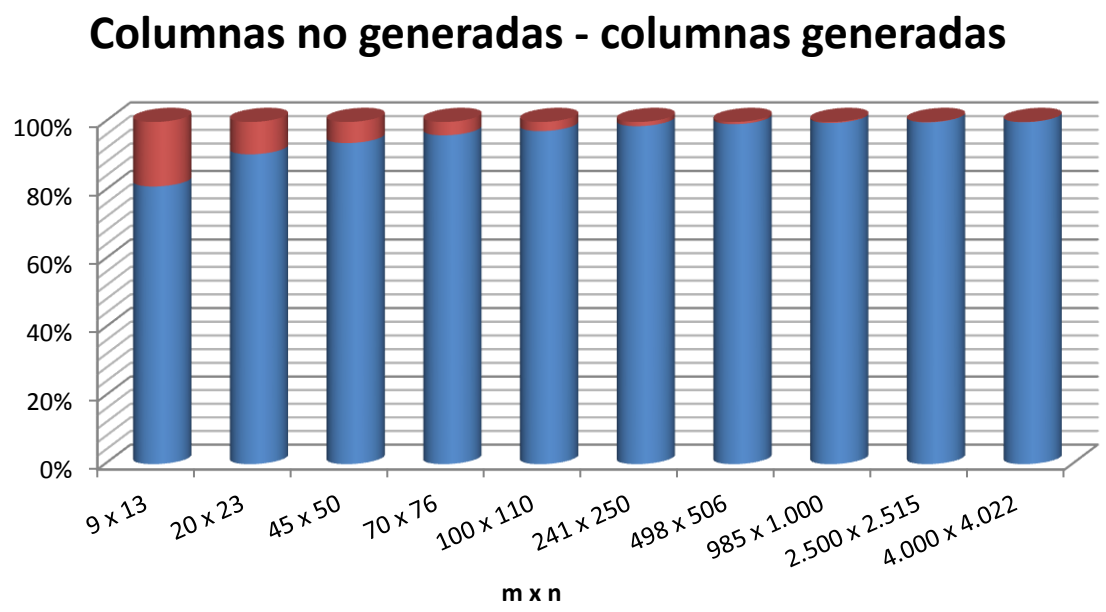

Gráfico 1

Como se puede además comprender, la utilización de un número de variables mucho menor, no solo facilita la resolución sino también el tiempo de ejecución del problema. No es lo mismo resolver un problema utilizando 16.088.000 variables, que resolver un problema en el que solamente se utilicen 8.988 variables.

3.2. Problema de asignación

3.2.1. Introducción al problema de asignación general

El problema de asignación consiste en asignar eficientemente una serie de actividades o tareas que tienen que ser realizadas por una serie de máquinas, personas, ordenadores… En general, se tendrá a m personas ($i = 1 \dots m$) que deben de realizar n actividades ($j = 1 \dots n$), no pudiendo ser divididas las actividades.

En el problema de asignación se tiene variables binarias $x_{ij} \in \{0, 1\}$, de tal forma que cuando la actividad j es asignada a la persona i, entonces $x_{ij} = 1$ y se tiene asociado un tiempo o costo de asignación c_{ij}.

Existen diferentes problemas de asignación, dependiendo del número de actividades que pueden ser desarrolladas por cada persona, aunque todos ellos tienen en común las restricciones de asignación:

$$\sum_{i=1}^{m} x_{ij} = 1; \;\; j = 1, \dots, n$$

El problema de asignación es uno de los problemas más estudiados, para cuya resolución pueden aplicarse diferentes algoritmos como la heurística Greedy (también conocido como el algoritmo miope), el método de penalización de Vogel, el método húngaro, el método simplex…

3.2.2. Problema de asignación clásico lineal

Para nuestro caso particular de aplicación, necesitamos tener balanceado el problema de asignación, conocido como el problema de asignación clásico.

En el problema de asignación clásico, se parte de que cada persona puede realizar únicamente una actividad, coincidiendo así el número de personas con las actividades a realizar, pudiendo reformular el modelo como sigue,

$$Minimizar \; \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}$$

$$Sujeto\ a \quad \sum_{i=1}^{N} x_{ij} = 1; \;\; j = 1, \dots, N$$

$$\sum_{j=1}^{N} x_{ij} = 1; \;\; i = 1, \dots, N$$

$$x_{ij} \in \{0, 1\}; \;\; i, j = 1, \dots, N$$

Donde tenemos a N personas $(j = 1, \dots, N)$ y N actividades $(i = 1, \dots, N)$, existiendo un costo o tiempo c_{ij} y una variable binaria x_{ij} para cada índice i, j, de tal forma que $x_{ij} = 1$, si se ha efectuado la asignación, y $x_{ij} = 0$, en caso contrario.

Aparentemente, nos encontramos ante un problema de programación entera. Debido a las propiedades especiales de la matriz de restricciones, específicamente al ser una matriz totalmente unimodular, es bien conocido que se pueden sustituir las restricciones de ser variables binarias por las de no-negatividad (ver Bazaraa-Jarvis capítulo 8) $(x_{ij} \geq 0)$. Ahora sí que nos encontraremos ante un problema de programación lineal de N^2 variables (tantas variables como columnas se podrían llegar a generar) y $2N$ restricciones, listo para ser resuelto mediante generación de columnas.

Como podemos observar, el modelo anterior se trata de una variante del problema del transporte, donde las ofertas y las demandas correspondientes a los orígenes y destinos respectivamente son 1, tal que $s_i = d_j = 1$. En el problema del transporte nos encontrábamos ante situaciones en las que uno o varios orígenes podían abastecer a uno o varios destinos, de igual modo que un destino podía ser abastecido por varios orígenes… Ahora, en el problema de asignación, nos encontraríamos en un caso especial, en el que cada origen alimenta a un único destino, y cada destino es abastecido únicamente por un único origen.

De ahora en adelante, nos referiremos al problema de asignación clásico lineal simplemente como el problema de asignación.

3.2.3. Resolución del problema de asignación mediante generación de columnas

Para llevar a cabo la resolución de un problema de programación lineal mediante generación de columnas, es necesario desarrollar un Problema Maestro Restringido (PMR), y a partir del cual, resolver un subproblema (P_ω). A continuación se desarrollará para el problema de asignación en particular.

Podemos considerar el Problema Maestro Restringido como aquel, formado por un conjunto G de columnas tal que sea posible llevar a cabo una asignación entre N personas y N actividades

$$\text{(PMR) Minimizar} \quad \sum_{(i,j)\in G} c_{ij}x_{ij}$$

$$\text{Sujeto a,} \quad \sum_{\{i:(i,j)\in G} x_{ij} = 1, \quad \mathrm{j} = 1, \dots, \mathrm{N}$$

$$\sum_{\{j:(i,j)\in G} x_{ij} = 1, \quad \mathrm{i} = 1, \dots, \mathrm{N}$$

$$\mathrm{x}_{ij} \geq 0, \quad \forall(\mathrm{i},\mathrm{j}) \in \mathrm{G}$$

Todo problema maestro restringido inicial tiene que cumplir la condición de que sea factible. Esa solución inicial puede consistir en tomar una solución mediante alguno de los métodos ya explicados en el problema del transporte, o mediante una solución tipo Greedy, eligiendo costes mínimos hasta completar las asignaciones. Otra alternativa puede consistir en hacer asignaciones sobre la diagonal principal tomando como soluciones las $x_{ii} = 1$, e igualando a cero el resto de variables. En cualquiera de los casos, tras resolver el Problema Maestro Restringido mediante el método simplex, se generaría una solución dual asociada a dos vectores duales $\vec{\omega}^1, \vec{\omega}^2 \in R^N$.
Para determinar si esa solución es óptima, es necesario calcular el coste reducido de la columna (i,j) como sigue,

$$z_{ij} - c_{ij} = \vec{\omega} a_{ij} - c_{ij} = \vec{\omega}_i^1 + \vec{\omega}_j^2 - c_{ij}$$

Y por lo tanto, en el subproblema habrá que calcular cual es el máximo coste reducido entre los elementos de la matriz

$$(P_\omega) \quad \underset{(i,j)}{Máx} \left\{ \vec{\omega}_i^1 + \vec{\omega}_j^2 - c_{ij} \right\} = \vec{\omega}_k^1 + \vec{\omega}_l^2 - c_{kl}$$

Si nos encontramos en una situación tal que $\vec{\omega}_\mathrm{k}^1 + \vec{\omega}_\mathrm{l}^2 - \mathrm{c_{kl}} \leq 0$, nos encontraremos ante la solución óptima. En el caso en el que $\vec{\omega}_\mathrm{k}^1 + \vec{\omega}_\mathrm{l}^2 - \mathrm{c_{kl}} > 0$, se habrá generado una nueva variable $\mathrm{x_{kl}}$, diciéndose que se ha generado una nueva columna, la cual se añadirá a G, volviendo a resolver el Problema Maestro Restringido con esa nueva columna.

3.2.4. Programación del problema de asignación mediante generación de columnas

A continuación se ha llevado a cabo una programación en Mosel, mediante el programa Xpress IVE, referente al problema de asignación mediante generación de columnas. Se ha establecido un número de 100 columnas, con dos criterios de parada; O bien tras encontrar la solución óptima, o bien, tras

un número determinado de iteraciones máximas. Como solución inicial, se ha propuesto la de asignaciones sobre la diagonal. El código fuente de la programación fue el siguiente;

```
model "Asignación con generación de columnas"
uses "mmxprs";
uses "mmsystem"
declarations   !declaramos las variables
 n = 100
 filas = 1..n
 costo:array(filas,filas) of integer
 w1:array(filas) of real
 w2:array(filas) of real
 final, ki, kj, iter:integer
 dmax:real
 tol = 0.001
 iter_max = 100000
 base: basis

 x:array(filas,filas) of mpvar
 ncol: integer
 costo_total:linctr

 salida_completa = 1

end-declarations


!Si queremos que se repitan los mismos resultados, activamos:
!setrandseed(12345)


!Generamos costos aleatorios
forall(i in filas,j in filas)costo(i,j):=round(round(100*random ))

write ("La matriz de costos es la siguiente:")
writeln


!Para ver por pantalla la matriz de costos
forall(i in filas)do
```

```
                writeln
        forall(j in filas)write("\t",costo(i,j))
end-do
writeln
writeln



!! Problema inicial:
ncol:=n
iter:=1
!Como solución inicial, tomaremos aquella en la que las x(i,i) = 1 (en la diagonal)
forall(j in filas)create(x(j,j))
costo_total:=sum(j in filas)costo(j,j)*x(j,j)
        forall(i in filas)res_fila(i):= x(i,i)=1
        forall(j in filas)res_col(j):= x(j,j)=1


! Bucle de generación de columnas:
starttime:= gettime
final:=0
repeat
        iter+=1
        minimize(costo_total)
        savebasis(base)

!Calculamos la solución dual
        forall(i in filas)do
    w1(i):=res_fila(i).dual
    w2(i):=res_col(i).dual
        end-do

!Buscamos la solución dual máxima
        dmax:=-99999
        forall(i,j in filas)do
          if(w1(i)+w2(j)-costo(i,j) > dmax) then
            dmax:=w1(i)+w2(j)-costo(i,j)
            ki:=i
            kj:=j
          end-if
        end-do
```

```
!Añadimos la variable /columna con solución dual máxima
        if(dmax <= tol)then final:=1
        else  ! aquí añado la nueva columna

                if(salida_completa=1)then
                  writeln("iteracion ",iter,", costo actual =",getobjval,
                          ", se va a annadir la columna (",ki,",",kj,")")
                end-if

                ncol+=1
                create(x(ki,kj))

                !Calculamos el Nuevo costo total
                costo_total+=costo(ki,kj)*x(ki,kj)

                res_fila(ki)+=x(ki,kj)
                res_col(kj)+=x(ki,kj)

                loadprob(costo_total)
                loadbasis(base)
   end-if


!Establecemos un criterio de parada
until (final =1 or iter >iter_max)


!Imprimimos resultados por pantalla
writeln("\nColumnas posibles= ",n*n,", columnas generadas = ",ncol)
writeln(" Columnas generadas (%) = ",100-100*(n*n-ncol)/(n*n) ," %")
writeln("\nn = ",n,", columnas generadas = ",ncol)
writeln("\n\nProblema actual. Costo total = ",getobjval)
writeln("\nTiempo total = ", gettime-starttime, " sec")

if(salida_completa=1)then
  writeln("\nSolucion optima:\n")
  forall(i,j in filas)
    if(x(i,j).sol>0)then writeln("x(",i,", ",j,") = 1")
    end-if

  writeln("\n\n")
```

```
  exportprob(0,"",costo_total)

end-if
end-model
```

3.2.5. Resultados experimentales. Problema de asignación

Mediante la programación del problema anterior, variando el parámetro n pueden resolverse problemas ficticios de tamaño tan grande como se desee. A continuación, vamos a analizar y comentar la salida del programa para un tamaño $n = 10$. Dicho análisis será también válido para mayores tamaños de n, tan solo dificultándose la inteligibilidad debido al mayor tamaño del problema.

Partiendo de una matriz de costos aleatoria, y de una solución inicial basada en la asignación de la diagonal, se van a ir introduciendo en nuestro problema nuevas columnas, dirigiéndose así hacia la solución óptima. La salida del programa Mosel tras la ejecución del código anterior para un $n = 10$ es la siguiente:

La matriz de costos es la siguiente:

6	*96*	*84*	*92*	*29*	*9*	*16*	*83*	*47*	*34*
32	*2*	*99*	*54*	*45*	*4*	*15*	*71*	*86*	*91*
55	*60*	*100*	*6*	*25*	*84*	*80*	*98*	*22*	*45*
91	*40*	*65*	*17*	*4*	*0*	*33*	*61*	*22*	*28*
52	*7*	*9*	*5*	*53*	*91*	*2*	*82*	*80*	*27*
91	*93*	*81*	*35*	*58*	*14*	*40*	*93*	*44*	*23*
43	*25*	*37*	*86*	*45*	*21*	*11*	*60*	*74*	*4*
26	*86*	*38*	*41*	*82*	*28*	*8*	*15*	*84*	*15*
64	*26*	*52*	*62*	*38*	*3*	*25*	*24*	*91*	*16*
18	*38*	*66*	*18*	*19*	*83*	*78*	*22*	*58*	*51*

iteracion 2, costo actual =360, se va a annadir la columna (5,3)
iteracion 3, costo actual =360, se va a annadir la columna (3,9)
iteracion 4, costo actual =360, se va a annadir la columna (9,5)
iteracion 5, costo actual =185, se va a annadir la columna (3,5)
iteracion 6, costo actual =185, se va a annadir la columna (9,10)
iteracion 7, costo actual =185, se va a annadir la columna (10,3)

iteracion 8, costo actual =185, se va a annadir la columna (10,5)
iteracion 9, costo actual =131, se va a annadir la columna (7,10)
iteracion 10, costo actual =131, se va a annadir la columna (8,7)
iteracion 11, costo actual =131, se va a annadir la columna (2,7)
iteracion 12, costo actual =131, se va a annadir la columna (1,7)
iteracion 13, costo actual =131, se va a annadir la columna (10,8)
iteracion 14, costo actual =131, se va a annadir la columna (7,3)
iteracion 15, costo actual =131, se va a annadir la columna (4,5)
iteracion 16, costo actual =131, se va a annadir la columna (3,4)
iteracion 17, costo actual =131, se va a annadir la columna (9,6)
iteracion 18, costo actual =131, se va a annadir la columna (6,9)
iteracion 19, costo actual =108, se va a annadir la columna (6,4)
iteracion 20, costo actual =108, se va a annadir la columna (9,8)
iteracion 21, costo actual =108, se va a annadir la columna (4,9)
iteracion 22, costo actual =108, se va a annadir la columna (6,10)
iteracion 23, costo actual =108, se va a annadir la columna (10,1)
iteracion 24, costo actual =108, se va a annadir la columna (10,4)

Columnas posibles= 100, columnas generadas = 35
Columnas generadas (%) = 35 %

Problema actual. Costo total = 108

Tiempo total = 0.101 sec

Solucion optima:

x(1, 1) = 1
x(2, 2) = 1
x(3, 4) = 1
x(4, 5) = 1
x(5, 3) = 1
x(6, 9) = 1
x(7, 10) = 1
x(8, 7) = 1
x(9, 6) = 1
x(10, 8) = 1

\ Using Xpress-MP extensions
Minimize
6 x(1,1) + 16 x(1,7) + 2 x(2,2) + 15 x(2,7) + 100 x(3,3) + 6 x(3,4) + 25 x(3,5) + 22 x(3,9) + 17 x(4,4) + 4 x(4,5) + 22 x(4,9) + 9 x(5,3) + 53 x(5,5) + 35 x(6,4) + 14 x(6,6) + 44 x(6,9) + 23 x(6,10) + 37 x(7,3) + 11 x(7,7) + 4 x(7,10) + 8 x(8,7) + 15 x(8,8) + 38 x(9,5) + 3 x(9,6) + 24 x(9,8) + 91 x(9,9) + 16 x(9,10) + 18 x(10,1) + 66 x(10,3) + 18 x(10,4) + 19 x(10,5) + 22 x(10,8) + 51 x(10,10)
Subject To

res_col(10): x(6,10) + x(7,10) + x(9,10) + x(10,10) = 1
res_col(9): x(3,9) + x(4,9) + x(6,9) + x(9,9) = 1
res_col(8): x(8,8) + x(9,8) + x(10,8) = 1
res_col(7): x(1,7) + x(2,7) + x(7,7) + x(8,7) = 1
res_col(6): x(6,6) + x(9,6) = 1
res_col(5): x(3,5) + x(4,5) + x(5,5) + x(9,5) + x(10,5) = 1
res_col(4): x(3,4) + x(4,4) + x(6,4) + x(10,4) = 1
res_col(3): x(3,3) + x(5,3) + x(7,3) + x(10,3) = 1
res_col(2): x(2,2) = 1
res_col(1): x(1,1) + x(10,1) = 1
res_fila(10): x(10,1) + x(10,3) + x(10,4) + x(10,5) + x(10,8) + x(10,10) = 1
res_fila(9): x(9,5) + x(9,6) + x(9,8) + x(9,9) + x(9,10) = 1
res_fila(8): x(8,7) + x(8,8) = 1
res_fila(7): x(7,3) + x(7,7) + x(7,10) = 1
res_fila(6): x(6,4) + x(6,6) + x(6,9) + x(6,10) = 1
res_fila(5): x(5,3) + x(5,5) = 1
res_fila(4): x(4,4) + x(4,5) + x(4,9) = 1
res_fila(3): x(3,3) + x(3,4) + x(3,5) + x(3,9) = 1
res_fila(2): x(2,2) + x(2,7) = 1
res_fila(1): x(1,1) + x(1,7) = 1

Bounds

End

Como podemos observar, inicialmente se genera una matriz cuadrada $n\ x\ n$ de costes aleatorios, que muestra los costes incurridos en que una actividad (por filas) sea realizada por una persona (por columnas).

Recordemos que como solución inicial, se había elegido la de asignación en la diagonal. A partir de ese momento (iteración 1), el programa va a ir introduciendo nuevas columnas (a la vez que va sacando otras) al problema, que nos van a permitir ir acercándonos hasta la solución final óptima. Para el caso que nos ocupa, se realizaron un total de 23 iteraciones, generándose un total de 33 columnas (teniendo en cuentas las diez generadas inicialmente).

Además, el programa también nos proporciona el coste total concurrido para la solución óptima el cual fue de 108 unidades monetarias, así como las 10 variables o columnas que hacen que la solución sea la óptima.

La salida del programa también nos muestra la función objetivo a minimizar, formada por las 33 columnas totales generadas, aunque recordemos que solo 10 de ellas tendrán valor 1. Dicha función objetivo estará sujeta a una

serie de restricciones por filas y por columnas, donde se tiene que cumplir que $\sum_{\{i:(i,j)\in G} x_{ij} = 1$ y que $\sum_{\{j:(i,j)\in G} x_{ij} = 1$.

Por último, mencionar el número de columnas que ha sido necesario generar para llegar al óptimo (se han generado 35 columnas), con respecto al número total de columnas que podrían haberse generado (podrían haberse generado hasta 100 columnas). En este caso, podemos observar que se han generado un 35% de las soluciones posibles.

Nótese en la salida del programa, como durante algunas iteraciones, la solución no mejora. Esto se debe a que el problema de asignación es un problema degenerado, lo que quiere decir, que pueden darse numerosas iteraciones en la que el algoritmo no mejora la solución existente.

Es necesario recordar, que el principal objetivo del algoritmo de generación de columnas, es el de llegar al óptimo de los problemas sin haber tenido en cuenta todas las variables. En el problema de asignación, aunque existen algoritmos combinatoriales que encuentran el óptimo incluso para problemas de grandes dimensiones, se ha querido aplicar la técnica de generación de columnas como una pequeña introducción a la generación de columnas, aunque el modelo completo funciona perfectamente hasta ciertas dimensiones.

Al igual que hicimos en el problema del transporte, a continuación, se muestra una tabla en la que se recogen los datos obtenidos tras diferentes ensayos, con diferentes tamaños.

Tamaño n	Columnas posibles	Columnas generadas	Columnas generadas (%)	Tiempo de ejecución (s)
10	100	35	35%	0,265
40	1.600	186	11,625 %	1,25
75	5625	381	6,7733 %	3,969
100	10.000	503	5,03 %	6,531
250	62.500	1.411	2,2576%	50,969
500	250.000	3.348	1,3392%	334,766
750	562.500	4.594	0,8167%	828,016
1.000	1.000.000	6.241	0,6241%	1.794,28
2.500	6.250.000	14.434	0,2309%	19.719,9

Tabla 5

Como podemos observar, a medida que incrementamos el tamaño de n, naturalmente se incrementa el número de variables o columnas posibles. Como es natural, el número de columnas generadas también se incrementa aunque en menor proporción.

Si nos fijamos en la cuarta columna, columnas generadas (%), podemos observar el porcentaje de columnas generadas, el cual, a medida que incrementamos el tamaño del problema va disminuyendo. Dicho porcentaje ha pasado de ser de un 35% para un tamaño de 10 x 10 a ser tan solo de un 0.231% para un tamaño de 2500 x 2500 variables.

Es aquí donde podemos apreciar la gran utilidad de la generación de columnas, pudiendo comprobar como de 6.250.000 variables posibles, solo ha necesitado generar 14.434 variables para encontrar la mejorar solución.

A continuación se muestran una gráfica donde se muestra de forma más visual los resultados recogidos en la tabla anterior. Para el 100% de columnas posibles, vemos la proporción de columnas que sí que han sido generadas y la proporción de columnas que no, pudiendo observar además como va variando ese porcentaje.

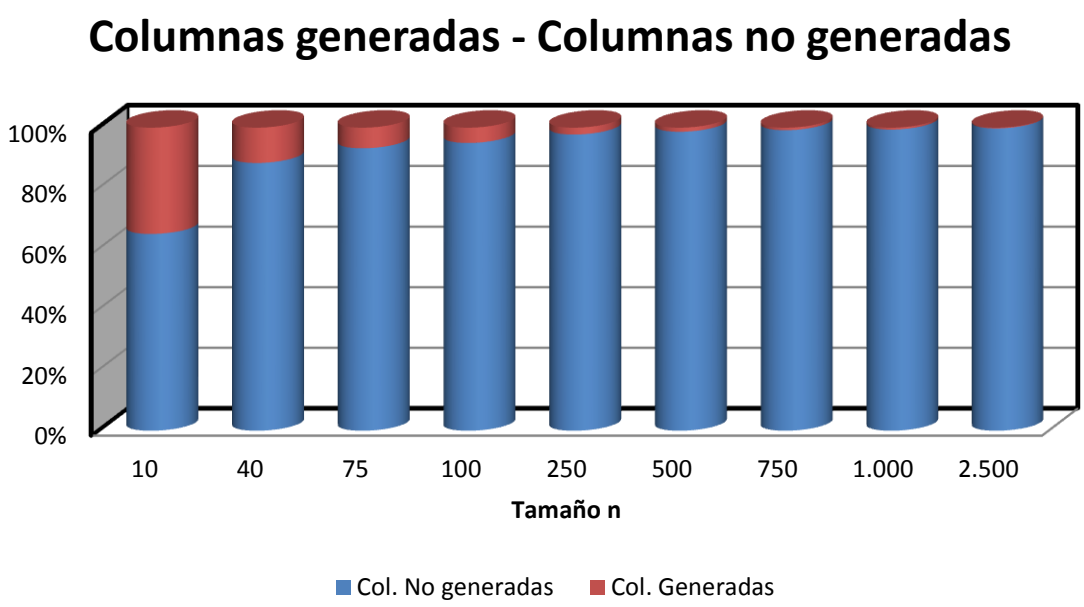

Gráfico 2

Capítulo 4. Programación de la producción de neumáticos.

4.1. Conceptualización

Durante esta última parte de mi trabajo fin de grado, me centraré en la programación del proceso de fabricación de neumáticos de una gran compañía, como podría ser Michelin, Bridgestone, Goodyear,... Estas compañías tienen que llevar a cabo la fabricación diaria de cientos de neumáticos, basándose no solo en el mismo tipo de neumático. Este proceso de producción puede abarcar desde neumáticos tan sencillos como los de bicicletas hasta neumáticos tan grandes y complejos como los elaborados para camiones o tractores. Y como se puede suponer, cada neumático tiene un distinto proceso de elaboración.

La elaboración de un neumático no solo consiste en la sencilla mezcla de gomas con una cocción final, sino en un elaborado proceso de fabricación. En función del tipo de neumático, cada uno de ellos utiliza distintas variedades de materias primas; diferentes tipos de cauchos, cables metálicos y textiles, productos químicos... Todo ello con el fin de dotar al neumático de unas prestaciones determinadas. Tras un cuidadoso proceso de mezcla, preparación y montaje del neumático, finalmente se requiere una última etapa, la de cocción.

La cocción permite que los materiales del neumático pasen del estado plástico al estado elástico, creándose así la estructura compuesta entre los diferentes elementos de la cubierta. Para ello, se utilizan unos moldes especiales que generan una presión sobre el neumático que ha sido colocado en su interior. Estos moldes son introducidos durante un determinado periodo de tiempo, que es denominado tiempo de cura, en calentadores que proporcionan el calor determinado para la cocción. La presión sirve para comprimir el neumático desde el interior y para aplicarla contra las paredes del molde para que tome la forma y estructura, durante el proceso de cocción. Como se puede suponer, cada neumático requiere de un tiempo de cocción determinado.

En lo que sigue, no nos centraremos en el proceso de fabricación completo de los distintos neumáticos, sino más bien en la última parte de este proceso, en la cocción. Para ello, partiremos de la idea de que podemos disponer de tantos neumáticos sin cocer como deseemos. Nuestra fabricación se verá determinada por las demandas de neumático y sus prioridades, las cuales

tendremos que maximizar, viéndose limitado por la disponibilidad de moldes y calentadores.

4.2. Aplicación a la generación de columnas

4.2.1. Visión global del problema.

Hay que maximizar la fabricación de diferentes tipos de neumáticos sujetos a una demanda, teniendo en cuenta que cada tipo de neumático tiene que ir asociado a un tipo de molde determinado, cuya disponibilidad es determinada. Posteriormente, estos moldes serán introducidos unos encima de otros en calentadores, hasta llenar el calentador, y teniendo en cuenta la disponibilidad de calentadores. Muy brevemente, este será el proceso del que nos vamos a ocupar. A continuación, empezaremos a verle en más detalle.

Dispondremos de diferentes demandas para cada tipo distinto de neumático. Además, cada demanda tendrá una prioridad determinada. Puede haber neumáticos cuya fabricación puede ser más urgente que otros, en términos de pedidos y prioridades. Esta prioridad será la que se intente satisfacer en la mayor medida de lo posible.

Cada neumático, tendrá unas dimensiones específicas; espesor, diámetro externo y diámetro interno. Otro parámetro importante de los neumáticos será el de los tiempos de cura, que indica el tiempo mínimo necesario que el neumático tiene que ser cocido para que adquiera las propiedades necesarias. Este parámetro se tendrá en cuenta durante la cocción. Las ruedas que estén en el mismo calentador tendrán que pertenecer al mismo grupo de cura. Un grupo de cura estará formado por neumáticos cuyo máximo y mínimo tiempo de cura no difiera en más de 100 minutos. Todos los neumáticos que cumplan esta condición, podrán cocerse a la vez durante el mismo tiempo, en el mismo calentador.

Del mismo modo, los moldes asociados tendrán también esas dimensiones específicas para cada tipo de neumático, aunque de distinto valor. Obviamente, el diámetro externo y el espesor del molde serán mayores que el del neumático, mientras que el diámetro interno del molde será inferior al del neumático.

Utilizando la siguiente ilustración, la cual va a ser utilizada para ilustrar las 3 dimensiones específicas, tanto de las ruedas como de los moldes. La cota 1 hace referencia al ancho de rueda del neumático o del molde, al cual nosotros nos referiremos de aquí en adelante como espesor. La cota 2 refleja

el diámetro interno de la rueda o del molde. Finalmente, la cota 3 hace referencia al diámetro externo de la rueda o del molde.

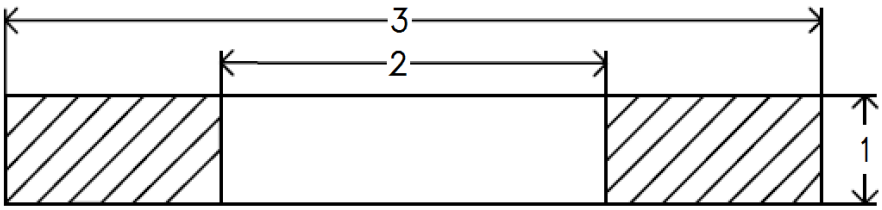

Ilustración 4

Además, otro parámetro de los moldes será el de la disponibilidad de cada tipo. El espesor del molde influirá en el número de moldes que entran en un determinado calentador. Los diámetros de los molden determinarán los requisitos de apilamiento en el calentador, ya que los moldes tienen que ir amontonados ordenadamente. No podrá situarse un molde encima de otro si el molde que está encima tiene un diámetro externo que es inferior al diámetro interno del molde que está debajo. De darse esa situación, el molde superior se colaría.

Una clara ilustración de esta situación queda reflejada en la ilustración 5, donde podemos observar como un conjunto de ruedas ha sido apiladas dentro del diámetro interno de una rueda. Este sería un claro ejemplo de incompatibilidad del apilamiento.

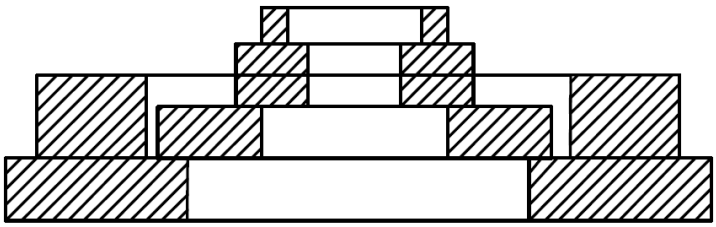

Ilustración 5

En estos casos, se utilizarán unas tablas metálicas espaciadoras entre esos moldes, con el fin de hacer soporte al molde superior y evitar que este se cuele. Dichas barras espaciadores, serán estándar para cualquier tipo de apilamiento. A continuación se muestra un factible apilamiento de ruedas utilizando una barra espaciadora.

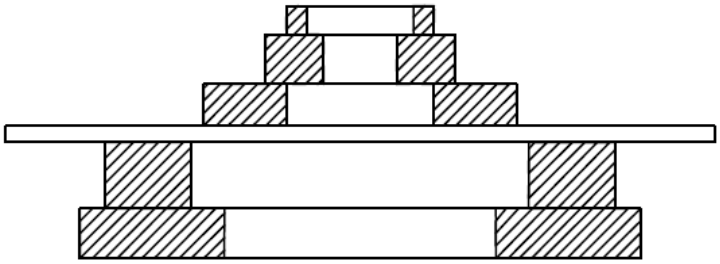

Ilustración 6

En cuanto a los calentadores, habrá una existencia determinada de estos, con diferentes magnitudes. La anchura de los calentadores es estándar y suficiente para que cualquier tipo de molde pueda entrar en ellos (en términos de diámetros externos de los moldes). En cuanto a la altura, esta podrá variar de unos tipos de calentadores a otros, pero teniendo en cuenta que siempre, en el calentador de menor altura, podrá entrar al menos uno de los moldes de mayor espesor.

4.2.2. Visión global de la generación de columnas.

Una grandísima variedad de soluciones podría ser generada para cada una de las posibles combinaciones de neumáticos que cumplieran con las restricciones y los requisitos del problema. Pero matemáticamente, sabemos que muchas de esas soluciones no nos permitirían alcanzar la solución óptima. Por ello, utilizaremos el algoritmo de generación de columnas, mediante el cual iremos generando las soluciones que nos van conduciendo hacia la solución óptima.

Enmarcando la generación de columnas a este problema, podemos asemejar una columna generada con una combinación factible de neumáticos con sus correspondientes moldes para un calentador determinado. La ilustración 6 mostrada anteriormente, reflejaba un apilamiento factible de las ruedas. Nótese que cada subproblema será específicamente diseñado para un determinado calentador y un grupo de cura.

Una combinación de neumáticos factibles implica:

1. Los neumáticos tienen que pertenecer al mismo tiempo de cura.
2. La suma de las alturas de los moldes donde se encuentran los neumáticos, no pueden sobrepasar la altura del calentador.
3. Los moldes tienen que ser colocados de una forma compatible.

Tras cumplirse las condiciones anteriores, la columna generada tienen que maximizar la prioridad promedia de los neumáticos que son producidos, sujeto a:

1. No se puede producir neumáticos que no han sido demandados. Se fabrica en base a necesidades.
2. No se pueden usar más moldes de los que disponemos, los cuales son limitados.
3. Al menos una columna es elegida para cada calentador. Esto quiere decir que no se puede dejar neumáticos sin fabricar teniendo calentadores sin usar. Al no ser, claro está, que no haya moldes disponibles.

Por lo tanto, la resolución de este problema constará de un proceso iterativo que partirá de una solución inicial, la cual tendrá que ser calculada previamente. Durante este proceso iterativo, se irán generando diversas columnas que irán mejorando la solución inicial. Este proceso de resolución constará de 3 pasos.

Paso 1. Resolución de un problema maestro como un problema lineal, usando las columnas generadas. Los precios duales o precios sombras nos proporcionarán información referente a la demanda de neumáticos atendida, al aprovechamiento de los moldes, y al aprovechamiento de los calentadores.

Paso 2. Resolución de un subproblema, a partir de la información obtenida de los precios duales. En este subproblema, se generarán columnas candidatas a encaminar el problema hacia la solución óptima. Para ello, se intentará en mayor medida de lo posible satisfacer las prioridades de demanda, controlar el uso de moldes y aprovechar al máximo los calentadores.

Paso 3. Si se encuentra una columna adecuada, entonces se volverá al paso 1 y se resolverá de nuevo el problema maestro.

4.3. El problema Maestro

4.3.1. Notaciones del problema maestro

Para comprender en detalle el problema maestro, a continuación se presentan los conjuntos, parámetros, índices y variables de decisión que participan en el código correspondiente al problema maestro.

A continuación se muestran los cuatro conjuntos que influyen en el código del problema maestro junto con sus correspondientes índices:

R Conjunto de tipo de ruedas (índice i)

C Conjunto de tipos de calentadores (índice k)

P Conjunto de prioridades (índice p)
Columnas Conjunto de columnas generadas (índice t)

Las variables que participan en el problema maestro son:
V_p Vector con el valor de las posibles prioridades
a_{it} Neumáticos tipo i en la columna t
c_{kt} =1, si la columna t es diseñada para el calentador k; 0 en cualquier otro caso.

demanda_ruedas $_{ip}$ Demanda de cada tipo de rueda organizada en prioridades
disp_molde $_i$ Disponibilidad de moldes
altura_calentador $_k$ Altura de cada tipo de calentador

Las variables de decisión son las siguientes:
U_{ip} Número de neumáticos tipo i producidos con prioridad p
W_t = 1, si la columna t es elegida, 0 en cualquier otro caso.

Los precios duales son los que siguen:
λ_i Referente a la prioridad de los neumáticos fabricados
μ_i Referente al uso de los moldes
v_k Referente al aprovechamiento de espacio en los calentadores.

Con este apartado tan solo se pretende hacer un pequeño acercamiento al problema maestro, ya que en el próximo apartado analizaremos en mayor detalle la función objetivo y cada una de las restricciones, junto con el estudio del modelo del problema maestro.

4.3.2. El modelo del problema maestro

A continuación se muestra la función objetivo junto con las restricciones correspondientes del problema maestro.

$$Maximizar \sum_{i=1}^{R}\sum_{p=1}^{P} V_p U_{ip}$$

$$Sujeto\ a:$$

$$Restricción\ 1 \quad \sum_{t=1}^{columnas} a_{it} * w_t = \sum_{p=1}^{P} u_{ip} \ ; \ \forall\, i \in R$$

$$Precio\ dual;\ \lambda_i$$

$$Restricción\ 2 \quad u_{ip} \leq demanda\_ruedas_{ip} \quad \forall\, i \in R$$

$$Restricción\ 3 \quad \sum_{t=1}^{columnas} a_{it} * w_t \leq Disp\_molde_i; \quad \forall\, i \in R$$

$$Precio\ dual;\ \mu_i$$

$$Restricción\ 4 \quad \sum_{t=1}^{columnas} c_{kt} * w_t \leq altura\_calentador_k; \quad \forall\, k \in C$$

$$Precio\ dual;\ v_k$$

$$Restricción\ 5 \quad w_t = \{0, 1\} \quad \forall\, t \in Columnas$$

La función objetivo será la de maximizar la prioridad media de fabricación de las ruedas, basada en las necesidades de la demanda. Para ello, habrá que tener en cuenta las cinco restricciones mostradas anteriormente.

La igualdad correspondiente a la restricción 1 nos indica que el sumatorio por columnas de los neumáticos producidos tipo i tiene que ser igual a la suma por prioridades de los neumáticos producidos tipo i. O dicho de otra forma, el número total de neumáticos tipo i fabricados con diferentes prioridades, tiene que ser igual al número total de neumáticos tipo i producidos. Esta restricción, lleva un precio sobra o precio dual asociado, λ_i, el cual será posteriormente utilizado en el subproblema.

La restricción 2 se basa en que la producción total de neumáticos no podrá ser nunca superior a la demanda de estos. El objetivo de esta restricción es la de evitar la sobreproducción, ya que no mejorará la función objetivo.

La no utilización de más moldes de los que se disponen según los datos, queda reflejado en la restricción 3. Obviamente, como ya mencionábamos en páginas anteriores, la empresa cuenta con una serie de moldes limitados para cada tipo de neumático, pudiéndonos encontrar en ocasiones con situaciones en la que nos falten moldes para poder satisfacer la demanda, y situaciones en las que nos sobren moldes. Esta restricción va acompañada de su correspondiente precio sombra μ_i.

De forma parecida, en la restricción 4 nos encontramos con la limitación de los calentadores. La empresa contará con diferentes modelos de calentadores, pudiendo haber varios ejemplares de cada uno. En ninguno de los casos, se podrá utilizar más calentadores de los que se dispone. Dicha

restricción también va acompañada de su precio dual, v_k, y cuyo desempeño explicaremos más adelante con más detalle junto con los anteriores.

Por último, la restricción 5, o restricción de integrabilidad, que hace referencia a la variable de decisión w_t, la cual valdrá 0 o 1 en función de si esa variable es elegida o no.

Para llegar a la resolución de este problema maestro, debemos realizar una relajación lineal. El problema maestro es resuelto como un problema continuo con variables reales, en vez de como un problema discreto, con variables enteras. Esto, implica realizar una relajación al problema en general y a la restricción 5 en particular, quedando esta como sigue,

$$0 \leq w_t \leq 1 \quad \forall\, t \in Columnas$$

Tras el replanteamiento de la restricción 5, la variable de decisión podrá tomar valores fraccionales. Esto afectará al valor de la función objetivo, cuya prioridad total del problema relajado será un valor un poco superior de la mejor solución de todas las columnas generadas en el problema real.

Resolviendo el problema maestro como un problema lineal, obtendremos los precios sombra de la restricción de los neumáticos (restricción 1) y de la restricción de los moldes (restricción 3). Estos precios duales, λ_i y μ_i, son los costes que necesitaremos en el subproblema para encontrar neumáticos/moldes para la nueva columna. Además, anteriormente, también mencionábamos el precio sombra v_k asociado a los calentadores. Este nos indicará el grado de aprovechamiento de espacio de los calentadores, de tal forma que, aquellos calentadores que tengan un mayor desaprovechamiento de espacio, serán aquellos que entren como nuevas columnas en el subproblema. El objetivo, será el de buscarles una mejor asignación de neumáticos y moldes para intentar llenar al máximo el calentador.

4.4. El subproblema

4.4.1. Notaciones del subproblema

El subproblema se encarga de generar nuevas e interesantes combinaciones de neumáticos para un determinado calentador y un determinado grupo de cura. Por ello, una columna es una combinación de neumáticos en sus correspondientes modelos y que además pertenecen a un mismo grupo de cura. Todo ello, para un determinado calentador. Esto tiene que ser tenido en

cuenta, puesto que solo los neumáticos que pertenecen al mismo grupo de cura, pueden ser curados a la vez en el mismo calentador.

Por consecuente, tendremos tantos subproblemas como combinaciones de grupos de cura y calentadores tengamos.

Para una mayor comprensión del subproblema cuando posteriormente se presente el modelo, a continuación se presentan los conjuntos, parámetros, índices y variables de decisión que intervienen en el subproblema.

Cabe recordar que cada subproblema está específicamente diseñado para un calentador y un grupo de cura determinado. Por ello, nos encontraremos con un único conjunto;

RP Conjunto de ruedas utilizadas para una columna determinada

En cuanto al resto de variables y constantes, son las que siguen,

S_{ip} Disponibilidad de ruedas organizadas por prioridades para un determinado grupo de cura

p_i Prioridad promedia del neumático tipo i

m_i Espesor del molde i

esp Constante relativa al espesor del espaciador

$altura$ Altura relativa al calentador del subproblema

mm_{ij} Matriz cuadrada i x j, referente a los requisitos de apilamiento de los neumáticos, cuyas posiciones pueden ser 3;
=2 si para colocar la rueda i encima de la rueda j se necesita un espaciador
=1 si la rueda i se puede colocar sin problema encima de la rueda j.
=0, en el caso en el que la rueda i no se pueda colocar encima de la rueda j (En el caso de que no sean del mismo grupo de cura, por ejemplo).

t_{0j} Referente a la rueda que va en la base del apilamiento

t_{1j} Referente a la rueda que va en la cima del apilamiento

Las variables de decisión son las siguientes:

y_i Número de neumáticos tipo i fabricados

z_j =1 si el molde tipo i es usado. =0 en cualquier otro caso.

y_{00} Variable que cuantifica el número de espaciadores usados

t_{ij} =1 si el molde i (o su correspondiente neumático i) está situado justo encima del molde j, con i ≠ j. o en cualquier otro caso

Por último, los precios sobras que ya aparecieron en el problema maestro

λ_i Referente a la prioridad de los neumáticos fabricados

μ_i Referente al uso de los moldes

Nótese, que v_k ya no está presente, debido a que es utilizada durante el problema maestro para determinar el calentador que es más conveniente optimizar.

4.4.2. El modelo del subproblema

A continuación se muestra la función objetivo del subproblema junto con las restricciones correspondientes.

$$Maximizar \sum_{i \in RP} (p_i - \lambda_i - \mu_i) * y_i$$

$Sujeto\ a$:

$$Restricción\ 1 \quad \sum_{i \in RP} y_i \leq \sum_{p} S_{ip} \ ; \ \forall\, i \in RP$$

$$Restricción\ 2 \quad esp * y_{00} + \sum_{i \in RP} m_i * y_i \leq altura$$

$$Restricción\ 3 \quad y_{00} = \sum_{i \in RP} \sum_{j \in RP \mid mm_{ij}=2} t_{ji}$$

$$Restricción\ 4 \quad \sum_{j \in RP} t_{0j} + \sum_{i \in RP \mid mm_{ij}\neq 0} t_{ij} \quad \forall\, j \in RP$$

$$Restricción\ 5 \quad \sum_{j \in RP} t_{1j} + \sum_{i \in RP \mid mm_{ij}\neq 0} t_{ji} \quad \forall\, j \in RP$$

$$Restricción\ 6 \quad \sum_{j \in RP} t_{0j} \leq 1$$

$$Restricción\ 7 \quad \sum_{j \in RP} t_{1j} \leq 1$$

$$Restricción\ 8 \quad z_j = \{0,1\} \quad \forall\, j \in RP$$

$$Restricción\ 9 \quad t_{0j},\ t_{1j},\ t_{ij} = \{0,1\} \quad \forall\, \mathrm{i}, j \in RP$$

La función objetivo será la de maximizar la prioridad media de fabricación de las ruedas, teniendo en cuenta para ello los precios sombra. Además, habrá que tener en cuenta las nueve restricciones mostradas anteriormente.

Comenzando por la primera restricción, se especifica que los neumáticos producidos para un determinado calentador, no pueden ser superiores a los neumáticos demandados organizados por prioridades. Si recordamos de la introducción a este problema en apartados anteriores, decíamos que no podíamos fabricar más de lo demandado.

La segunda restricción, se trata de una restricción tipo mochila. En este caso, la mochila se trata de un calentador, y lo que se pretende con esta restricción es que los elementos que van incluidos dentro del calentador no superen bajo ningún concepto la altura de este. O lo que es lo mismo, que la suma de los espaciadores usados más los moldes asignados a ese calentador en particular no sea superior a la altura del calentador.

El número de espaciadores usados es controlado mediante la restricción 3. Para ello, se recorre la matriz cuadrada mm, de tal forma que si en la intersección de la fila i con la columna j hay un 2, es señal de que hay que utilizar un espaciador entre esas ruedas.

Las restricciones 4 y 5 son utilizadas para forzar la condición del apilamiento piramidal que veíamos en la ilustración 6 mostrada anteriormente. Esta restricción de secuenciación de los moldes implica que si un molde está siendo utilizado, entonces tiene que estar situado encima de otro molde (restricción 4) y del mismo modo, tiene que tener encima de él otro molde (restricción 5). Para completar los requisitos de apilamiento, nótese que solo las variables t_{ij} relevantes son tenidas en cuenta, utilizando para ello la matriz mm anteriormente mencionada.

La restricción 6 y la restricción 7 respectivamente son utilizadas para los moldes postizos. Esto quiere decir que utilizamos la variable t_{0j} como un molde falso que está situado en la base del apilamiento de los moldes. Del mismo modo, la variable t_{1j} es utilizada como un molde falso que está situado en la cima de la columna generada. Por ello, mediante estas dos

restricciones se impone la condición de tener un molde falso situado en la base y otro en la cima de cada columna generada.

La restricción 8, se trata de una condición de no negatividad e integrabilidad de la variable z_j cuyo valor será 1 si el molde j está siendo utilizado, o valdrá 0 en caso contrario.

Finalmente la restricción 9 impone otra condición de integrabilidad y no negatividad, en este caso a las variables t_{0j}, t_{1j}, t_{ij}, cuya función ya ha sido explicada con anterioridad.

Desde un punto de vista global, el subproblema se trata de un problema real y general tipo mochila, pero algo más complejo debido a la utilización de las condiciones de secuenciación.

El subproblema se ejecutará alternativamente junto con el problema maestro, durante un continuo proceso de intercambio de datos entre ambos programas, como ya se mencionó en el capítulo anterior. Durante este proceso iterativo, el subproblema generará columnas referentes a nuevos calentadores, con el fin de optimizar su uso, y aprovechar el espacio de estos al máximo. Todo ello, guiado a optimizar la producción de neumáticos y maximizar las prioridades.

Durante este proceso iterativo, los precios sombras que mencionábamos anteriormente irán variando. A medida que se consigan tener calentadores con un gran aprovechamiento del espacio, los costes reducidos irán tendiendo a cero. Acercándose así hacia el óptimo del problema, momento en el que finalizará el problema.

Una vez comprendido el modelo tanto del problema maestro, como del subproblema, y el funcionamiento del proceso de fabricación de ruedas, en el próximo apartado se muestra la implementación de dicho modelo, utilizando para ello el lenguaje de programación Mosel, junto con el posterior análisis de resultados.

4.5. Estructura algorítmica

4.5.1. Notaciones generales del algoritmo.

Para una mejor comprensión, a continuación se muestran todos los tipos de notaciones que han sido utilizadas en la implementación del problema que hemos visto en previos apartados.

R Conjunto de tipo de ruedas (índice i)
C Conjunto de tipos de calentadores (índice k)
P Conjunto de prioridades (índice p)

Columnas Conjunto de columnas generadas (índice t)
n_grupo Conjunto de grupos de cura
V_p Vector con el valor de las posibles prioridades
D_ruedas_i Diámetro externo de las ruedas
d_ruedas_i Diámetro interno de las ruedas
$ancho\_ruedas_i$ Espesor de las ruedas
t_curado_i Tiempo mínimo de curado requerido para cada rueda
$demanda\_ruedas_{ip}$ Demanda de cada tipo de rueda organizada en prioridades
$demanda\_total_i$ Demanda total de cada tipo de rueda
$prioridad\_media_i$ Valor promedio de prioridad para cada tipo de ruedas

ncal Número de calentadores usados en la solución final
$disp\_molde_i$ Disponibilidad de moldes
$ancho\_molde_i$ Ancho de cada tipo de molde
D_molde_i Diámetro externo de cada molde
d_molde_i Diámetro interno de cada molde
mm_{ii} Matriz cuadrada sobre los requisitos de apilamiento de las ruedas

$altura\_calentador_k$ Altura de cada tipo de calentador
$uso\_calentador_k$ Vector para controlar la fabricación sujeto a demandas y disponibilidades de moldes
$disp\_calentador_k$ Ejemplares disponibles para cada tipo de calentador
$altura\_usada_k$ Vector para controlar el espacio usado para cada calentador
$tipo\_cal_k$ Variable usada para controlar el calentador elegido en el subproblema

a_{it} Neumáticos tipo i en columna t
b_{it} Moldes tipo i en columna t
c_{kt} =1, si la columna t es diseñada para el calentador k; 0 en cualquier otro caso.
$índice_i$ Vector que contiene los índices de las ruedas que van a participar en el subproblema

Las variables de decisión son las siguientes:
U_{ip} Número de neumáticos tipo i producidos con prioridad p
W_t = 1, si la columna t es elegida, 0 en cualquier otro caso.

Los precios sombras son los siguientes:
Lambda (λ_i) Referente a la prioridad de los neumáticos fabricados
mu(μ_i) Referente al uso de los moldes
nu(v_k) Referente al aprovechamiento de espacio en los calentadores.

4.5.2. Notaciones específicas del subproblema.

Como se sabe, el funcionamiento del problema consiste en un proceso iterativo en el cual se produce un intercambio continuo de datos entre el problema maestro y el subproblema. Esto hace que las variables, parámetros y conjuntos utilizados en el subproblema cambien de nombre. Otras, en cambio, son solo utilizadas en el subproblema. A continuación se presentan todas esas notaciones.

ruedas_sub $_i$ ***Conjunto de ruedas que participan en el subproblema***

índice_sub $_i$ Vector que contiene los índices de las ruedas que van a participar en el subproblema
altura Variable que indica la altura del calentador.
esp Constante que indica el espesor del espaciador

agen Para controlar la columna generada
cr Para controlar el coste reducido
y00 Para controlar el número de espaciadores utilizados

prior_sub $_i$ Vector que contiene las prioridades promedio de las ruedas que participan en el subproblema
dem_sub $_i$ Vector que informa sobre la demanda cubierta en el subprobema
ancho_sub$_i$ Vector que informa sobre el diámetro externo de los moldes utilizados en el subproblema
mm_sub $_{i\,i}$ Matriz que establece los requisitos de apilamiento de las ruedas

t0 $_j$ Referente a la rueda que va en la base del apilamiento
t1 $_j$ Referente a la rueda que va en la cima del apilamiento

y $_i$ Número de neumáticos tipo i fabricados

z_j =1 si el molde tipo i es usado. =0 en cualquier otro caso.
t_{ij} =1 si el molde i (o su correspondiente neumático i) está situado justo encima del molde j, con i ≠ j. o en cualquier otro caso

Y finalmente, los precios sombra son los siguientes:
lambda_sub (λ_i) Referente a la prioridad de los neumáticos fabricados
mu_sub (μ_i) Referente al uso de los moldes
n_sub (v_k) Referente al aprovechamiento de espacio en los calentadores.

4.5.3. Código fuente del problema maestro.

```
model "Problema de ruedas"
uses "mmxprs", "mmjobs", "mmsystem"

 declarations
        R, P, C :integer                    ! Número de piezas
 end-declarations

 initializations from "datos_ruedas_1.dat"
  R P C
 end-initializations


!=======================================================!
!================== DECLARACIÓN DE VARIABLES ===========!
!=======================================================!

declarations
        ruedas = 1..R
        calentadores = 1..C
        prioridades = 1..P

        D_ruedas: array(ruedas) of real !Medidas en mm
        d_ruedas: array(ruedas) of real
        ancho_ruedas: array(ruedas) of real
        t_curado: array(ruedas) of real !Expresado en minutos
        demanda_ruedas: array(ruedas, prioridades) of real !Necesidades de
             cada tipo de ruedas en función de prioridades
        demanda_total: array(ruedas) of real !Necesidades totales de cada tipo
             de ruedas
```

```
prioridad_media: array(ruedas) of real !Valor promedio de prioridad
    para cada tipo de ruedas
v:array(prioridades)of real          !valor de las posibles prioridades
ncal:array(calentadores)of integer ! número de calentadores usados
    en la solución final

!Cada tipo de ruedas tiene un molde asociado
disp_molde:array (ruedas) of integer
ancho_molde:array (ruedas) of real
D_molde:array (ruedas) of real
d_molde:array (ruedas) of real
mm:array (ruedas, ruedas) of integer

!Variables referentes a los calentadores
altura_calentador:array (calentadores) of real
uso_calentador:array (calentadores) of real
disp_calentador:array(calentadores)of real


!Variables usadas para la solución inicial
a1,d1, diam_ant, Diam_ant,esp:real
uso_molde:array(ruedas) of real

!Variables usadas para la ordenación de los tiempos de cura
t_min, t_max, maximo, parada, dif: real
n_grupo, nrp:integer
grupo_cura:array(range, ruedas) of integer
lambda:array(ruedas)of real
mu:array(ruedas)of real
nu:array(calentadores)of real

columnas: range              ! Conjunto de columnas
w: dynamic array(columnas) of mpvar     ! variables de decisión
u:array(ruedas,prioridades)of mpvar
tipo_cal:array(columnas)of integer
altura_usada:array(columnas)of real
ncol:integer
zpass:array(range)of real

a:array(ruedas,columnas)of integer          ! número de ruedas de tipo i
    en la columna t
```

```
        b:array(ruedas,columnas)of integer          ! número de moldes de tipo i
            en la columna t
        c:array(calentadores,columnas)of integer ! =1 si la columna t usa el
            calentador k
        npass_max = 100
        tolcr = 0.001

        !==================== subproblema ====================!
        subproblema:Model
        indice:array(ruedas)of integer
        altura:real

        estatus_sub:integer
        agen:array(ruedas)of integer
        agenmax:array(ruedas)of integer
        cgen:array(ruedas)of integer
        cgenmax:array(ruedas)of integer

        cr, crmax:real
        kmax:integer
        zini, zact:real
        tolz = 0.0001

        tiempo_max = 100

end-declarations

initializations from "datos_ruedas_1.dat"
 D_ruedas
 d_ruedas
 ancho_ruedas
 t_curado
 demanda_ruedas
 v
 disp_molde
 altura_calentador
 disp_calentador
 esp

end-initializations
```

```
!================================================ =====!
!================== ALGUNOS CÁLCULOS INICIALES ============!
!=========================================================!

!A partir de las necesidades de ruedas organizadas en prioridades,
!calculamos las necesidades globales de cada una, y calculamos la prioridad
!media de cada ruedas

forall(i in ruedas)demanda_total(i):=sum(p in prioridades)demanda_ruedas (i,p)
foral (i in ruedas) prioridad_media(i):= sum(p in prioridades) v(p) *
demanda_ruedas (i,p) / demanda_total (i)


!Creamos las características de los moldes en función de las características
de las ruedass
!Cada ruedas tiene un molde asociado
forall (i in ruedas) ancho_molde(i):=1.07*ancho_ruedas(i)
forall (i in ruedas) D_molde(i):=1.02*D_ruedas(i)
forall (i in ruedas) d_molde(i):=0.98*d_ruedas(i)
writeln("Dimensiones de los moldes:\n")
writeln("i\tancho\tdiamE\tdiamI\n")
forall (i in ruedas)write ("\n",i,"\t",ancho_molde(i),"\t",D_molde(i) ,"\t",
d_molde(i),"\n")
writeln

t1:=gettime


!=========================================================!
!========= ==== ORDENACIÓN DE LOS TIEMPOS DE CURA ===========!
!=========================================================!

!Comenzamos buscando el menor tiempo de cura t_min
writeln("\nGrupos de curado:\n")
writeln
t_min:=1000
n_grupos:=0
forall (i in ruedas)do
        if(t_min>t_curado(i))then
        t_min:=t_curado(i)
        end-if
end-do
```

```
!También vamos a buscar el mayor tiempo de cura, puesto que lo
!utilizaremos posteriormente como criterio de parada 'maximo'
maximo:=0
forall (i in ruedas)do
        if(t_curado(i)>maximo)then
        maximo:=t_curado(i)
        end-if
end-do

!Cada grupo de cura, estará delimitado por un t_min y un t_max
t_max:=t_min+100

!Iniciamos el bucle de ordenación de los tiempos de cura
while(parada<2)do
        !creamos el grupo de cura
        n_grupo+=1
        forall (i in ruedas)do
                if(t_curado(i)>=t_min and t_curado(i)<=t_max)then
                grupo_cura(n_grupo,i):=1
                end-if
        end-do

        !Imprimimos resultados por pantalla
        write("Grupo de cura ", n_grupo, ":  ")
        forall(i in ruedas)write(grupo_cura(n_grupo,i)," ")
        writeln
        !Buscamos el nuevo t_min
        dif:=1000
        ind:=0
        forall(i in ruedas)do
                if(t_curado(i)-t_min<dif and t_curado(i)>t_min)then
                dif:=(t_curado(i)-t_min)
                ind:=i
                end-if
        end-do

        !Actualizamos datos
        t_min:=t_curado(ind)
        t_max:=t_min+100

        !Criterio de parada
```

```
        if(t_max>=maximo)then
        parada+=1
        end-if

end-do
writeln
writeln("Numero de grupos de cura = ",n_grupo," // maximo = ", maximo," //
t_max = ", t_max )
writeln
forall(k in 1..n_grupo)do
  writeln
  forall( i in ruedas)write(grupo_cura(k,i),"  ")
end-do
writeln


!=====================================================!
!============ ============= Matriz mm ====================!
!=====================================================!

forall(i in ruedas, j in ruedas)do !Rueda i encima de rueda j
        forall(k in 1..n_grupo)do
                if(grupo_cura(k,i)=1 and grupo_cura(k,j)=1)then
                    !Son del mismo grupo de cura
                                if(D_molde(i)<d_molde(j)or D_molde(j)<d_molde(i) )
                                then  !La rueda de encima se cuela -->  Espaciador
                                  mm(i,j):=2
                                  else !No se necesita espaciador
                                  mm(i,j):=1
                                end-if
                end-if
        end-do
end-do

writeln
writeln
writeln("Matriz mm: ")
forall(i in ruedas)do
  writeln
  forall(j in ruedas)write(mm(i,j),"  ")
end-do
writeln
```

```
writeln


!=========================================================!
!============== ======= SOLUCIÓN INICIAL ==================!
!=========================================================!
writeln("\nSolución inicial:\n")
ncol:=0
!Primero determinamos las necesidades totales de cada tipo de ruedas a
partir de las prioridades
!forall (i in ruedas)demanda_total(i):=sum(j in prioridades)demanda_ruedas(i,j)
writeln("Necesidades totales de ruedass:")
forall (i in ruedas)write(demanda_total(i),"\t")
writeln

!Ahora la adjudicación ruedas a moldes
forall (i in ruedas)do
        if(demanda_total(i)<disp_molde(i)) then
          uso_molde(i):=demanda_total(i)
        else
          uso_molde(i):=disp_molde(i)
        end-if
end-do
writeln
writeln("Moldes que se van a usar:")
forall (i in ruedas) write (uso_molde(i),"\t")
writeln
writeln

(!Adjudicación moldes a calentadores
Esto nos proporcionará la solución inicial. Lo que haremos será
adjudicar los moles a los calebtadores sin tener en cuenta prioridades
ni nada de eso, tan solo adjudicando desde las ruedass tipo 1 hasta
que se acabe el espacio disponible.!)
b1:=1

forall (k in calentadores)do
        if(b1<=R)then
                d1:=disp_calentador(k)  !Para controlar el numero de
calentadores de cada tipo
                writeln
                writeln
```

```
        writeln ("Calentador tipo ",(k),": ")
else
        writeln
        writeln
        write ("El calentador tipo ",(k)," no es usado ")
end-if

while(d1>=1 and b1<=R)do
        d1-=1
        diam_ant:=0
        Diam_ant:=10000000
        a1:=0 !Para controlar la altura del calentador y no superarla
        c1:=0 !Para cambiar de calentador
        writeln
        write ("    ejemplar ",d1+1,": ")

        while(c1=0 and b1<=R)do
                if(uso_molde(b1)<=0)then
                        b1+=1
                end-if

                uso_molde(b1)-=1
                a1+=ancho_molde(b1)

                if(altura_calentador(k)>=a1  and  diam_ant<=D_molde(b1)
                and Diam_ant>=d_molde(b1))then
                        a(b1,ncol)+=1
                        b(b1,ncol)+=1
                        c(k,ncol):=1
                        tipo_cal(ncol):=k
                        write ("ruedas ",(b1),"  ")
                        !actualizamos los datos para la próxima iteracción
                        Diam_ant:=D_molde(b1)
                        diam_ant:=d_molde(b1)
                end-if

                !Hemos sobrepasado la altura del calentador
                if(altura_calentador(k)<a1)then
                        c1:=1
                        uso_molde(b1)+=1
                        a1-=ancho_molde(b1)
                end-if
```

```
!La ruedas se cuela --> Espaciador !!
!Si la ruedas se cuela utilizamos un espaciador
!Pero si al intentar usar un espaciador la ruedas no entra
!lo intentamos con otra ruedas --> b+1

!Diam_ant < d_molde(b1)
if(Diam_ant<d_molde(b1) and c1=0)then
        a1+=esp

        if(altura_calentador(k)>=a1)then
                write ("espaciador  ruedas ",(b1),"  ")
                a(b1,ncol)+=1
                b(b1,ncol)+=1
                c(k,k):=1
                Diam_ant:=D_molde(b1)
                diam_ant:=d_molde(b1)
        else
                uso_molde(b1)+=1
                a1-=ancho_molde(b1)
                a1-=esp
                b1+=1
        end-if
end-if

!diam_ant > D_molde(b1)
if(diam_ant>D_molde(b1) and c1=0)then
        a1+=esp

        if(altura_calentador(k)>=a1)then
                write ("espaciador  ruedas ",(b1),"  ")
                a(b1,ncol)+=1
                b(b1,ncol)+=1
                c(k,k):=1
                Diam_ant:=D_molde(b1)
                diam_ant:=d_molde(b1)
        else
                uso_molde(b1)+=1
                a1-=ancho_molde(b1)
                a1-=esp
                b1+=1
        end-if
```

```
                    end-if


                    !No quedan moldes tipo 'b'
                    if(uso_molde(b1)<=0)then
                        b1+=1
                    end-if

                end-do !Del segundo do-while
                ncol:=ncol+1
                create(w(ncol))
                w(ncol)is_binary
                !w(k)is_integer

        end-do !Del primer do-while
end-do !Del forall

writeln
writeln
write("Matriz a:\n")
forall(i in ruedas)do
  writeln
  forall(t in columnas)write(a(i,t)," ")
end-do

writeln
writeln
write("\nMatriz b:\n")
forall(j in ruedas)do
        writeln
        forall(t in columnas)write(b(j,t)," ")
end-do
writeln

writeln("Solución inicial con ",getsize(w)," columnas")
writeln
writeln

!Ruedas propuestas para fabricación segun solucion inicial
writeln("Ruedas propuestas para fabricación segun solución inicial")
forall(t in columnas)write(sum(i in ruedas)a(i,t), " ")
```

```
!Aprovechamiento de espacio en la solución inicial
writeln
writeln("\nUso de los calentadores en la solución inicial:\n")
forall(t in columnas|tipo_cal(t)>0)do
 altura_usada(t):=sum(i in ruedas)ancho_molde(i)*a(i,t)
 writeln(" columna ",t+1,", tipo calentador = ",tipo_cal(t)," // Altura
    disponible = ",altura_calentador(tipo_cal(t)) ,", Altura usada =
    ",altura_usada(t))
 ncal(tipo_cal(t))+=1
end-do



!=====================================================!
!*****************************************************!
!********************** PROBLEMA MAESTRO ***************!
!*****************************************************!
!=====================================================!

forall(i in ruedas,p in prioridades)u(i,p)is_integer

prioridad_total:=sum(i in ruedas,p in prioridades)v(p)*u(i,p)

forall(i in ruedas )res1(i):= sum (t in columnas) a(i,t) * w(t) = sum (p in
prioridades) u(i,p)

forall(i in ruedas,p in prioridades)res2(i,p):= u(i,p)<=demanda_ruedas(i,p)

forall(i in ruedas)res3(i):= sum(t in columnas)a(i,t)*w(t)<=disp_molde(i)

forall(k in calentadores) res4(k) :=  sum (t in columnas) c(k,t) * w(t) <=
disp_calentador(k)

writeln("\nProblema Maestro inicial:")
!exportprob(EP_MAX,"",prioridad_total)
maximize(XPRS_LIN,prioridad_total)

!=====================================================!
!============== Se compila y se carga el subproblema ============!
!=====================================================!
if compile("subproblema.mos")<>0 then        ! Para compilar el subproblema
        writeln("Algún problema al compilar subproblema.mos")
```

```
            exit(1)
end-if

load(subproblema, "subproblema.bim")       ! Se carga el subproblema

!=========================================================!
!================= GENERACIÓN DE COLUMNAS ================!
!=========================================================!

defcut:=getparam("XPRS_CUTSTRATEGY") ! Save setting of `CUTSTRATEGY'
setparam("XPRS_CUTSTRATEGY", 0)     ! Disable automatic cuts
setparam("XPRS_PRESOLVE", 0)        ! Switch presolve off
!setparam("zerotol", eps)           ! Set comparison tolerance of Mosel

npass:=0
final:=0

zini:=-999.9
while(final=0 and npass<=npass_max) do
    npass:=npass+1

    if(npass>1)then zini:=getobjval;end-if

    maximize(XPRS_LIN,prioridad_total)

    zpass(npass):=getobjval
          writeln("\n  w(ncol) = ",w(ncol).sol)
          zact:=getobjval

    if(zact - zini < tolz)then
      final:=1

    else

          forall(i in ruedas)lambda(i):=getdual(res1(i))
          forall(j in ruedas)mu(j):=res3(j).dual
    forall(k in calentadores)nu(k):=res4(k).dual
          !writeln("\n\n  lambda = ",lambda)
          !writeln("  mu = ",mu)
          !writeln("  nu = ",nu)
    writeln("\n**Pasada ",npass,", prioridad_total = ",getobjval)
```

```
!forall(t in columnas)writeln("w(",t,") = ",w(t).sol)

      crmax:=-9999.9

forall(k in calentadores, f in 1..n_grupo)do
  ! Aquí hay que ejecutar el subproblema
  ! Número posible de ruedas en el subproblema

  forall(i in ruedas)do
    if(grupo_cura(f,i)=1)then
      indice(i):=1
    else
      indice(i):=0
    end-if
  end-do
  altura:=altura_calentador(k)

  initializations to "shmem:datos1"
   R
        indice
        prioridad_media
        lambda
        mu
        altura
        esp
        ancho_molde
        demanda_total
        disp_molde
        mm
  end-initializations

  run(subproblema)
            wait                        ! Wait until subproblem finishes
            dropnextevent                   ! Ignore termination message

            initializations from "shmem:datos_sub"
                  agen
                  cr
                  estatus_sub
            end-initializations

            cr:=cr-nu(k)
```

```
                !writeln("  k = ",k , ";   nu=   ", nu(k),", f = ",f," ==> cr = ",cr)
                if(cr > crmax)then
                  crmax:=cr
                  forall(i in ruedas)agenmax(i):=agen(i)
                  kmax:=k
                end-if

            if(cr  > tolcr)then
            ncol:=ncol+1
                    create(w(ncol))
                    w(ncol)is_binary
                    !w(ncol)is_integer
                    forall(i in ruedas|agen(i)>0)do
                      res1(i)+=agen(i)*w(ncol)
                      res3(i)+=agen(i)*w(ncol)
                      a(i,ncol):=agen(i)
                    end-do
                    tipo_cal(ncol):=k
                    res4(k)+=w(ncol)

                end-if

    end-do  ! final de forall(k in calentadores, f in 1..n_grupo)

    if(crmax  <= tolcr)then
                final:=1
          end-if
         end-if
end-do  ! final de while(final=0 and npass<=npass_max)

writeln("\n**Solución LP final: z = ",getobjval,", ncol = ",ncol)
write("\nRuedas producidas:")
forall(i in ruedas)write(sum(p in prioridades)u(i,p).sol,",")
writeln

(!writeln("\nMatriz a y valores de w:\n")
forall(i in ruedas)do
  writeln
  forall(t in columnas|w(t).sol>0.01)write(a(i,t),"\t")
end-do
writeln!)
```

```
writeln
writeln
writeln("\nValores de w:\n")
writeln
forall(t in columnas)write(strfmt(w(t).sol,5,3),"\t")
writeln

writeln("\ntiempo total en LP: ",gettime-t1," segundos")
!exportprob(EP_MAX,"",prioridad_total)

!=========================================================!
!======== ============= Fase 2 (branch and cut) ===============!
!=========================================================!
setparam("XPRS_MAXTIME",tiempo_max)
maximize(prioridad_total)
writeln("\n\n**Solución PE final: z = ",getobjval)
write("\n  ruedas producidas:")
forall(i in ruedas)write(sum(p in prioridades)u(i,p).sol,",")
writeln("\nUso de los calentadores:\n")
forall(t in columnas|w(t).sol>=0.99 and tipo_cal(t)>0)do
  altura_usada(t):= sum(i in ruedas)ancho_molde(i)*a(i,t)
writeln("  columna ",t,", tipo calentador = ",tipo_cal(t),"  //   Altura disponible =
      ",altura_calentador(tipo_cal(t)) ,",  Altura usada = ",altura_usada(t))
  ncal(tipo_cal(t))+=1
end-do
writeln("ncal = ",ncal)
writeln("Columnas generadas: ",ncol)

mip_status:= getparam("XPRS_MIPSTATUS")
mejor_cota:= getparam("XPRS_BESTBOUND")
nodos:=getparam("XPRS_NODES")
if(getobjval<>0)then
   hueco_final:= 100*(mejor_cota-getobjval)/getobjval
end-if

writeln("\nMIP Estatus = ",mip_status)
writeln("\nNodos examinados = ",nodos)

writeln("\n***Solucion final")
writeln("Valor objetivo= ",getobjval)
writeln("Mejor cota= ",mejor_cota)
writeln("Hueco final= ",hueco_final," %")
```

```
writeln("\ntiempo total: ",gettime-t1," segundos")

!Aprovechamiento de espacio en la solución final
writeln("\nMatriz a definitiva:\n")
forall(i in ruedas)do
  writeln
  forall(t in columnas|w(t).sol>0.01)write(a(i,t),"\t")
end-do
writeln

end-model
```

4.5.4. Código fuente del subproblema.

```
model ModelName
uses "mmxprs"; !gain access to the Xpress-Optimizer solver

declarations
        R:integer
end-declarations

initializations from "shmem:datos1"
        R
end-initializations

!writeln("Desde el subproblema: R = ",R)

declarations
        ruedas = 1..R
        indice:array(ruedas)of integer
        demanda_total:array(ruedas)of integer
        disp_molde:array(ruedas)of integer
        prioridad_media:array(ruedas)of real
        lambda, mu:array(ruedas)of real
        altura, esp:real
        ancho_molde:array(ruedas)of real

        ind_sub:array(ruedas)of integer
        agen:array(ruedas)of integer
        estatus_sub:integer
```

```
        mm:array(ruedas,ruedas)of integer
end-declarations

initializations from "shmem:datos1"
        indice as 'indice'
        demanda_total as 'demanda_total'
        prioridad_media as 'prioridad_media'
        lambda as 'lambda'
        mu as 'mu'
        altura as 'altura'
        esp as 'esp'
        ancho_molde as 'ancho_molde'
        disp_molde as 'disp_molde'
        mm as 'mm'
end-initializations

! Número posible de ruedas en el subproblema y sus índices
nrp:=0
forall(i in ruedas)do
  if(indice(i)=1)then
     nrp:=nrp+1
     ind_sub(nrp):=i
  end-if
end-do

declarations
        ruedas_sub=1..nrp
        prior_sub:array(ruedas_sub)of real
        lambda_sub:array(ruedas_sub)of real
        mu_sub:array(ruedas_sub)of real
        n_sub:array(ruedas_sub)of integer          ! moldes desponibles
        dem_sub:array(ruedas_sub)of integer
        ancho_sub:array(ruedas_sub)of real
        mm_sub:array(ruedas_sub, ruedas_sub)of real

        y00:mpvar
        y:array(ruedas_sub)of mpvar
        z:array(ruedas_sub)of mpvar
        t:array(ruedas_sub,ruedas_sub)of mpvar
        t0:array(ruedas_sub)of mpvar
        t1:array(ruedas_sub)of mpvar
```

```
end-declarations

forall(i in ruedas_sub)do
  prior_sub(i):=prioridad_media(ind_sub(i))
  lambda_sub(i):=lambda(ind_sub(i))
  mu_sub(i):=mu(ind_sub(i))
  dem_sub(i):=demanda_total(ind_sub(i))
  n_sub(i):=disp_molde(ind_sub(i))
  ancho_sub(i):=ancho_molde(ind_sub(i))
  forall(j in ruedas_sub)do
        mm_sub(i,j):=mm(ind_sub(i),ind_sub(j))
  end-do
end-do

y00 is_integer
forall(i in ruedas_sub)y(i)is_integer
forall(i in ruedas_sub)z(i)is_binary
forall(i,j in ruedas_sub)t(i,j)is_binary
forall(i in ruedas_sub)t0(i)is_binary
forall(i in ruedas_sub)t1(i)is_binary

prioridad_sub:=sum(i in ruedas_sub)(prior_sub(i)-lambda_sub(i)-mu_sub(i))*y(i)

forall(i in ruedas_sub)res1(i):= y(i)<=minlist(dem_sub(i),n_sub(i))*z(i)
res2:= esp*y00+sum(i in ruedas_sub)ancho_sub(i)*y(i)<= altura
res3:= y00=sum(i in ruedas_sub,j in ruedas_sub|mm_sub(i,j)=2)t(j,i)

forall(j in ruedas_sub)res4(j):= t0(j)+sum(i in ruedas_sub|mm_sub(j,i)<>0)t(i,j)
forall(j in ruedas_sub)res5(j):=  t1(j)+sum(i in ruedas_sub|mm_sub(i,j)<>0)t(j,i)
res6:= sum(j in ruedas_sub)t0(j)<=1
res7:= sum(j in ruedas_sub)t1(j)<=1

!exportprob(EP_MAX,"",prioridad_sub)
maximize(prioridad_sub)
estatus_sub:=getprobstat

cr:=getobjval

if(getprobstat=XPRS_INF)then
    writeln("SubProblema no factible")
elif(getprobstat=XPRS_OPT or getprobstat=XPRS_UNF)then
```

```
   !writeln("En el subproblema, estatus = ",estatus_sub,", prioridad =
        ",getobjval)
   forall(i in ruedas_sub)agen(ind_sub(i)):=round(y(i).sol)

   initializations to "shmem:datos_sub"
    agen
    cr
    estatus_sub
   end-initializations
else
    writeln("Algún problema con la resolución del subproblema")
end-if

end-model
```

4.5.5. Datos utilizados.

Los datos utilizados para la ejecución del ejemplo que será analizada posteriormente, han sido tomados de un fichero externo. A continuación se adjuntan dichos datos, con el fin de poder facilitar la compresión de los resultados posteriores.

```
R:15
P:4
C:10
D_ruedas:
        [386,691,612,455,620,658,441,723,566,659,984,483,546,825,1097]
d_ruedas:
        [305,610,457,356,356,483,330,483,381,508,584,305,381,483,584]
ancho_ruedas:
        [135,135,155,165,165,175,185,185,205,215,235,255,275,285,285]
t_curado:
        [200,220,280,270,330,300,290,395,350,415,490,430,455,530,610]
demanda_ruedas:[
5,0,4,10,
0,4,1,0,
20,2,2,4,
5,0,0,8,
5,5,5,0,
0,4,10,4,
3,3,3,3,
3,0,0,17,
```

4,12,0,0,
10,10,7,0,
0,0,0,40,
6,0,15,0,
2,2,2,2,
10,2,3,4,
9,0,0,9]

v:[50,30,20,40]
disp_molde:[10,3,19,6,9,7,10,19,19,23,30,9,2,15,12]
disp_calentador:[5,4,3,2,3,4,4,8,2,2]
esp:177.8 !Tamaño en mm del espaciador
altura_calentador:[575,700,800,2800,800,700,1500,1250,1500,2000]

4.6. Análisis de resultados

A continuación vamos a ir comentando las partes más interesantes de la salida del programa.

Tal y como se dijo en apartados anteriores, cada neumático tiene un molde asociado. Las dimensiones específicas de dicho molde (véase ilustración 4) son calculadas por el programa a partir de las dimensiones específicas del neumático. Estas dimensiones específicas son: Diámetro externo, diámetro interno y ancho. A continuación se muestras dichas características recogidas en la siguiente tabla.

Rueda tipo	Ancho molde	Diámetro Externo	Diámetro interno
Tipo 1	144.45	393.72	298.9
Tipo 2	144.45	704.82	597.8
Tipo 3	165.85	624.24	447.86
Tipo 4	176.55	464.1	348.88
Tipo 5	176.55	632.4	348.88
Tipo 6	187.25	671.16	473.34
Tipo 7	197.95	449.82	323.4
Tipo 8	197.95	737.46	473.34
Tipo 9	219.35	577.32	373.38
Tipo 10	230.05	672.18	497.84
Tipo 11	251.45	1003.68	572.32
Tipo 12	272.85	492.66	298.9
Tipo 13	294.25	556.92	373.38
Tipo 14	304.95	841.5	473.34
Tipo 15	304.95	1118.94	572.32

Tabla 6

Algo de lo que también se encarga el programa es de elaborar los grupos de cura. Recordemos que las ruedas que pertenecen al mismo grupo de cura son aquellas que pueden ser cocidas al mismo tiempo en el mismo calentador. Para que dos ruedas puedan ser del mismo grupo de cura, el tiempo de cocción entre ellas no puede exceder los 100 minutos de diferencia. A continuación se muestra una tabla donde se informa de los diversos grupos de cura que han sido elaborados para los datos introducidos. Por filas, se muestran los distintos grupos de cura. Por columnas, se muestran los distintos tipos de rueda. Si en la intersección entre la fila i y la columna j hay un 1, eso quiere decir que la rueda j pertenece al grupo de cura i. En caso contrario, no. Nótese que una rueda puede pertenecer a varios grupos de cura.

	Tipos de rueda														
Grupos de cura	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
2	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0
3	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0
4	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0
5	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0
6	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0
7	0	0	0	0	1	0	0	1	1	1	0	1	0	0	0
8	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0
9	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0
10	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0
11	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
12	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
13	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Tabla 7

Otra matriz que se ha mencionado anteriormente y que también es calculada por el programa, es la matriz mm. Esta matriz es la encargada de llevar a cabo un adecuado ordenamiento de las ruedas en los diferentes calentadores. Se trata de una matriz cuadrada en función de los diferentes tipos de ruedas. Lo que se pretende con esta matriz es establecer los criterios de apilamiento de las ruedas.

Para una correcta interpretación de la matriz mm, y por lo tanto de la tabla 6 mostrada a continuación, hay que interpretar la matriz como si la fila i fuese colocada encima de la columna j. O lo que es lo mismo, como si la rueda tipo i es colocada encima de la rueda j. Entonces, partiendo de esta base, se comienzan a analizar los resultados.

→ Si la rueda tipo i que está situada encima de la rueda tipo j no son del mismo grupo de cura, entonces estas ruedas no son del mismo grupo de cura y por consiguiente, no se pueden apilar ya que no se pueden cocer a la vez. En este caso, en la intersección entre la fila i y la columna j aparecerá un 0.

→Partiendo de que la rueda i y la rueda j son del mismo grupo de cura, pueden darse dos situaciones distintas:

a) La rueda i puede ir colocada encima de la rueda j sin ningún problema. En este caso, aparecerá un 1 en la intersección entre fila i y columna j.

b) La rueda i se cuela entre la rueda j. Estaríamos en un caso de incompatibilidad de apilamiento, tal y como veíamos en la

ilustración 5. La solución, consiste en la de utilizar una barra espaciadora como la utilizada en la ilustración 6. En esta situación, habrá un 2 en la intersección entre fila i y columna j.

Tipo rueda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	2	1	0	2	1	0	0	0	0	0	0	0	0
2	2	1	1	2	0	1	2	0	0	0	0	0	0	0	0
3	2	1	1	1	1	1	1	0	1	0	0	0	0	0	0
4	1	2	1	1	1	2	1	0	1	0	0	0	0	0	0
5	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0
6	2	1	1	2	1	1	2	1	1	0	0	0	0	0	0
7	1	2	1	1	1	2	1	0	1	0	0	0	0	0	0
8	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0
9	0	0	1	1	1	1	1	1	1	1	0	1	0	0	0
10	0	0	0	0	1	0	0	1	1	1	1	2	1	0	0
11	0	0	0	0	0	0	0	1	0	1	1	2	2	1	0
12	0	0	0	0	1	0	0	1	1	2	2	1	1	1	0
13	0	0	0	0	0	0	0	1	0	1	2	1	1	1	0
14	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Tabla 8

Pueden comprobarse manualmente los datos anteriores, tan solo teniendo en cuenta las columnas 3 y 4 de la tabla 4 mostrada anteriormente, junto con la utilización de los grupos de cura de la tabla 5.

En cuanto a la solución inicial, esta ha sido programada de tal forma que se va rellenando los calentadores comenzando con el tipo de neumático 1. Cuanto se ha satisfecho toda la demanda correspondiente a dicho tipo de neumático, se pasa al siguiente tipo. Todo ello, tan solo teniendo en cuenta la demanda existente y no teniendo en cuenta las prioridades de las diferentes demandas, ni mucho menos el aprovechamiento de los calentadores.

A continuación se muestra la solución inicial elaborada. Para cada uno de los ejemplares de los diferentes calentadores, se muestra la secuencia de ruedas que son cocidos en ellos. Además, también se indica la utilización de espaciadores y su posición en caso de que estos hayan sido utilizados.

Calentador tipo 1:

Ejemplar 5: ruedas 1 ruedas 1 ruedas 1

Ejemplar 4: ruedas 1 ruedas 1 ruedas 1

Ejemplar 3: ruedas 1 ruedas 1 ruedas 1
Ejemplar 2: ruedas 1 espaciador ruedas 2
Ejemplar 1: ruedas 2 ruedas 2 ruedas 3

Calentador tipo 2:
Ejemplar 4: ruedas 3 ruedas 3 ruedas 3 ruedas 3
Ejemplar 3: ruedas 3 ruedas 3 ruedas 3 ruedas 3
Ejemplar 2: ruedas 3 ruedas 3 ruedas 3 ruedas 3
Ejemplar 1: ruedas 3 ruedas 3 ruedas 3 ruedas 3

Calentador tipo 3:
Ejemplar 3: ruedas 3 ruedas 3 ruedas 4 ruedas 4
Ejemplar 2: ruedas 4 ruedas 4 ruedas 4 ruedas 4
Ejemplar 1: ruedas 5 ruedas 5 ruedas 5 ruedas 5

Calentador tipo 4:
Ejemplar 2: ruedas 5 ruedas 5 ruedas 5 ruedas 5 ruedas 5 ruedas 6 ruedas 6 ruedas 6 ruedas 6 ruedas 6 ruedas 6 ruedas 6 espaciador ruedas 7 ruedas 7
Ejemplar 1: ruedas 7 ruedas 7 ruedas 7 ruedas 7 ruedas 7 ruedas 7 ruedas 7 ruedas 7 espaciador ruedas 8 ruedas 8 ruedas 8 ruedas 8 ruedas 8

Calentador tipo 5:
Ejemplar 3: ruedas 8 ruedas 8 ruedas 8 ruedas 8
Ejemplar 2: ruedas 8 ruedas 8 ruedas 8 ruedas 8
Ejemplar 1: ruedas 8 ruedas 8 ruedas 8 ruedas 8

Calentador tipo 6:
Ejemplar 4: ruedas 8 ruedas 8 ruedas 9
Ejemplar 3: ruedas 9 ruedas 9 ruedas 9
Ejemplar 2: ruedas 9 ruedas 9 ruedas 9
Ejemplar 1: ruedas 9 ruedas 9 ruedas 9

Calentador tipo 7:
Ejemplar 4: ruedas 9 ruedas 9 ruedas 9 ruedas 9 ruedas 9 ruedas 9
Ejemplar 3: ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10
Ejemplar 2: ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10
Ejemplar 1: ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10

Calentador tipo 8:

Ejemplar 8: ruedas 10 ruedas 10 ruedas 10 ruedas 10 ruedas 10
Ejemplar 7: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 6: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 5: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 4: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 3: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 2: ruedas 11 ruedas 11 ruedas 11 ruedas 11
Ejemplar 1: ruedas 11 ruedas 11 ruedas 11 ruedas 11

Calentador tipo 9:

Ejemplar 2: ruedas 11 ruedas 11 espaciador ruedas 12 ruedas 12 ruedas 12
Ejemplar 1: ruedas 12 ruedas 12 ruedas 12 ruedas 12 ruedas 12

Calentador tipo 10:

Ejemplar 2: ruedas 12 ruedas 13 ruedas 13 ruedas 14 ruedas 14 ruedas 14
Ejemplar 1: ruedas 14 ruedas 14 ruedas 14 ruedas 14 ruedas 14 ruedas 14

Si plasmamos toda la solución inicial anterior en una matriz, obtenemos la matriz a, que hemos mencionado durante la elaboración del modelo. Esta matriz muestra por columna cada uno de los 37 ejemplares de todos los calentadores, o lo que es lo mismo, las 37 columnas iniciales de las que partimos en el programa.

3 3 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 4 4 4 4 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 2 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 5 4 4 4 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 3 3 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 6 5 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4 4 4 4 4 4 4 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 5 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Realizando un resumen de la anterior matriz, y agrupando las columnas por calentadores, obtenemos la siguiente tabla, donde se recogen las ruedas que son propuestas para fabricación con la solución inicial.

Calentador tipo	1	2	3	4	5	6	7	8	9	10
Número ruedas	14	16	12	27	12	12	24	33	10	12

Tabla 9

En el problema del transporte y en el problema de asignación generábamos una nueva columna en cada iteración o pasada. Esta columna, coincidía con la de mayor coste reducido. En cambio, en este problema, vamos a generar en cada pasada o iteración todas aquellas columnas cuyo coste reducido sea mayor que cero. A continuación se muestran las iteraciones que han sido realizadas y el valor de la función objetivo para la relajación lineal del problema. El número de columnas generadas entre las siguientes iteraciones ha sido de 737.

**Pasada 1, prioridad_total = 6738
**Pasada 2, prioridad_total = 7600.03
**Pasada 3, prioridad_total = 7803.41
**Pasada 4, prioridad_total = 7815.29
**Pasada 5, prioridad_total = 7815.4
**Solución LP final: z = 7815.4
Número de columnas generadas = 737

Como sabemos, la solución del problema relajado no tiene por qué ser entera. Esto hace que esta solución sea una cota superior del problema real, el cual podemos observar a continuación, que presenta un valor de la función objetivo un poco inferior al conseguido en la relajación lineal.

**Solución PE final: z = 7750

Es en este momento cuando las variables de decisión w(t) dejan de ser reales y pasan a ser binarias, comienza a funcionar el algoritmo de Branch and cut en el problema de programación entera. Este algoritmo utiliza las columnas generadas en la parte lineal.

El pequeño empeoramiento afecta a las ruedas reales fabricadas. A continuación se muestra una tabla donde se comparan las ruedas fabricadas según la relajación lineal (RL) con las ruedas fabricadas en el problema real (PE).

Tipo Rueda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ruedas RL	10	3	19	6	9	7	10	19	16	21.27	30	6	2	15	12
Ruedas PE	10	2	19	6	9	7	10	19	15	21	30	6	2	15	12

Tabla 10

A continuación se muestra una tabla donde se compara el aprovechamiento de los calentadores entre la solución inicial y la solución final. Podemos ver por filas los distintos tipos de calentadores, mientras que por columnas, vemos la altura de la que disponen y su uso tanto en la solución inicial (SI) como en la solución final (SF).

Tipo calentador	Altura disponible	Altura usada (SI)	Altura usada (SF)
Tipo calentador 1	575	433.35	433.35
Tipo calentador 1	575	433.35	529.65
Tipo calentador 1	575	433.35	545.7
Tipo calentador 1	575	288.9	556.4
Tipo calentador 1	575	454.75	572.45
Tipo calentador 2	700	663.4	690.15
Tipo calentador 2	700	663.4	690.15
Tipo calentador 2	700	663.4	658.05
Tipo calentador 2	700	663.4	690.15
Tipo calentador 3	800	684.8	797.15
Tipo calentador 3	800	706.2	791.8
Tipo calentador 3	800	706.2	754.35
Tipo calentador 4	2800	2589.4	2589.4
Tipo calentador 4	2800	2573.35	2782
Tipo calentador 5	800	791.8	797.15
Tipo calentador 5	800	791.8	791.8
Tipo calentador 5	800	791.8	786.45
Tipo calentador 6	700	615.25	658.05
Tipo calentador 6	700	658.05	658.05
Tipo calentador 6	700	658.05	690.15
Tipo calentador 6	700	658.05	690.15
Tipo calentador 7	1500	1316.1	1471.25
Tipo calentador 7	1500	1380.3	1487.3
Tipo calentador 7	1500	1380.3	1487.3
Tipo calentador 7	1500	1380.3	1492.65
Tipo calentador 8	1250	1150.25	1219.8
Tipo calentador 8	1250	1005.8	1230.5
Tipo calentador 8	1250	1005.8	1219.8
Tipo calentador 8	1250	1005.8	1219.8
Tipo calentador 8	1250	1005.8	1235.85
Tipo calentador 8	1250	1005.8	1241.2
Tipo calentador 8	1250	1005.8	1219.8
Tipo calentador 8	1250	1005.8	1235.85
Tipo calentador 9	1500	1321.45	1481.95
Tipo calentador 9	1500	1364.25	1492.65
Tipo calentador 10	2000	1776.2	1829.7
Tipo calentador 10	2000	1829.7	1990.2

Tabla 11

Finalmente, por agrupar algunos datos que han sido mostrados en diversas tablas y de diferente forma, a continuación se muestra una tabla donde se puede observar la diferencia de ruedas fabricadas entre la solución inicial (SI) con respecto a las de la solución final (SF).

Ruedas SI	10	3	19	6	9	7	10	19	16	23	30	9	2	2	9
Ruedas SF	10	2	19	6	9	7	10	19	15	21	30	6	2	15	12
Diferencia	0	-1	0	0	0	0	0	0	-1	-2	0	-3	0	+13	+3

Tabla 12

Como podemos observar, se han dejado de fabricar 7 ruedas de diferentes tipos que antes sí que se fabricaban. A cambio, y junto con un mejor aprovechamiento del espacio, se han fabricado 16 ruedas que antes no se fabricaban. Si analizamos la diferencia, 9 ruedas a mayores han sido fabricadas. Y las buenas noticias no terminarían aquí, sino que también habría que tener en cuenta que la fabricación se ha basado en base a prioridades, habiéndose fabricado las de mayor prioridad pudiendo proporcionar unos mayores beneficios a la compañía.

CONCLUSIONES

En este trabajo fin de grado nos hemos propuesto llevar a cabo un acercamiento al método de generación de columnas con el fin de llevar a cabo su aplicación a diversos problemas logísticos. Entre ellos, el problema del transporte, el problema de asignación, y la optimización de un proceso productivo de fabricación de neumáticos.

Mientras que los dos primeros problemas presentan un modelo completo que funciona correctamente hasta ciertas dimensiones, el segundo de ellos no. Por ello, tanto para el problema de asignación como el del transporte no sería de gran utilidad aplicarles la generación de columnas, en cuanto a modelos pequeños se refiere pero sí para modelos de mayores dimensiones. Aun así, se ha llevado a cabo dicha aplicación para comprobar su funcionamiento y analizar los resultados.

La técnica de generación de columnas es ideal para tratar problemas de programación lineal de gran escala, como puede ser el proceso de fabricación de neumáticos, donde las diferentes combinaciones de moldes a calentadores pueden ser muy elevadas. Dicho algoritmo se basa en la aplicación del método simplex, pero sin generar todas las columnas posibles, sino solamente las de mayor coste reducido.

Para la resolución de los problemas, como ya mencionamos anteriormente, se lleva a cabo primeramente una resolución de la relajación lineal del problema. Durante esta parte de resolución se asume que la variable de decisión w (t) es continua y puede tomar cualquier valor entre 0 y 1. Posteriormente, durante la resolución del problema entero, se impondrá que esas variables tienen que ser binarias. En su implementación en la fase de generación de columnas, se pueden llevar a cabo distintas implementaciones en referente al número de columnas a adicionar. En cada iteración se puede añadir la columna que presente el mayor coste reducido, o por el contrario, se pueden añadir todas las columnas que presenten un coste reducido positivo.

La primera de esas implementaciones ha sido aplicada en el problema del transporte y en el problema de asignación, en los cuales, por cada iteración solamente añadimos la columna con mayor coste reducido. Esta alternativa puede presentar el problema del tiempo de ejecución, el cual puede verse incrementado. Además, el tiempo también puede verse afectado por el elevado número de columnas que pueden ser necesarias añadir.

Pese al posible tiempo de ejecución que se pueda tardar, conviene destacar nuevamente la gran utilidad de este algoritmo. La no necesidad de conocer todas las columnas. Fijándonos en los datos obtenidos del problema del transporte, en el caso particular del problema de 16.088.00 variables, sería imposible desarrollar una solución a mano, por no mencionar la elevadísima necesidad de memoria que sería necesaria para generar y calcular todas esas variables, además del tiempo que costaría llevarlo a cabo. Así, mediante la generación de columnas, podemos resolver el problema generando tan solo 8.988 variables.

Lo mismo ocurría con el problema de asignación, ya que tal y como vimos en los ejemplos, de 6.250.000 variables, tan solo hemos tenido que generar 14.434 para alcanzar una solución óptima.

Continuando con la aplicación al problema de asignación, si observamos la salida del programa, podemos ver que durante varias iteraciones, se añaden columnas las cuales no mejoran la función objetivo. Esto se debe a que el problema de asignación es un problema degenerado, que es un caso típico en problemas combinatorios.

En cambio, en el problema de optimización del proceso de fabricación de neumáticos, hemos seguido la segunda estrategia que mencionábamos anteriormente. En este caso, en cada iteración o pasada se añaden todas las columnas con coste reducido positivo. A la vista de los resultados para unos datos determinados, se han generado 737 columnas en tan solo 5 pasadas o iteraciones.

La resolución del problema relajado no tiene por qué ser una solución entera, lo que puede afectar en cierto modo a la solución del problema real. Esto se aprecia muy bien en el problema de los neumáticos, donde podemos observar como el grado de satisfacción, medido mediante la función objetivo, es mayor en la relajación lineal, que en el problema entero donde se queda con un valor un poco menor.

En el ejemplo que veíamos en el anterior apartado, podíamos observar cómo la función objetivo de la relajación lineal valía 7815.4, mientras que la función objetivo del problema entero era de 7750. También veíamos esa diferencia en las ruedas fabricadas en la relajación lineal y en el problema entero (véase tabla 10).

La técnica de generación de columnas es muy utilizada, tanto para problemas con el objetivo de hallar la solución exacta, como para obtener una solución aproximada del problema. Aunque si bien es dicho, hay otras técnicas que nos permiten obtener una solución aproximada en una resolución de tiempo

mucho más breve de lo que hoy en día nos puede ofrecer la generación de columnas.

A lo largo de este trabajo fin de grado hemos intentado plasmar las ventajas de utilizar la generación de columnas, no solo por la facilidad de poder resolver un problema sin tener que generar implícitamente todas las variables, sino también, porque pueden darse situaciones en las que no sea posible generar todas las alternativas. O bien porque hay columnas que se desconocen, o bien por una posible falta de memoria o por motivos de tiempo.

Tras la realización de dicho trabajo fin de grado, creo que las investigaciones en referente al tema de la generación de columnas, han experimentado numerosos e importantes avances durante los últimos años, hasta el punto de permitirnos llevar a cabo la resolución de problemas sin contar con todas las variables. Sin duda alguna, se han llevado a cabo grandes desarrollos que permiten la optimización de sistemas y procesos, tanto del sector servicios como del sector productivo, mediante la programación matemática. Algunas de estas aplicaciones pueden ser encontradas en el libro de Column Generation de Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon, donde también se recogen algunos de los últimos avances en referente al procedimiento de generación de columnas.

Aun así, esta técnica debería continuar siendo estudiada y perfeccionada con el fin de ampliar sus aplicaciones, agilizar sus resoluciones y conseguir una mayor estabilidad del algoritmo.

Bibliografía

1. Programación lineal y flujo en redes. Mokhtar S. Bazaraa, John J. Jarvis. Editorial Limusa. Año 1989.

2. Column generation. Guy Desaulniers, Jacques Desrosiers, Marius M. Solomon. Editorial Springer. Año 2005.

3. Apuntes de programación lineal, el problema del transporte, el problema de asignación y la generación de columnas elaborados y proporcionados por Jesús Sáez Aguado.

4. Apuntes de clase de la asignatura de Métodos cuantitativos en ingeniería de organización I.

5. A Tire Scheduling System for Bridgestone/Firestone Off-The-Road. Zeger Degraeve, Linus Schrage. Operation Research, vol.45, No. 6, November – December 1997.
http://pubsonline.informs.org/doi/pdf/10.1287/opre.45.6.789
http://www.lindo.com/index.php?view=article&id=95%3Aarchives-lindo-cures-big-tire-problems&option=com_content&Itemid=34

6. Manual Xpress ive. Mosel. (http://home.deib.polimi.it/malucell/didattica/appunti/mosel/mosel-language_reference.pdf).

7. Programación a gran escala: Andrés L. Medaglia, PH.D. (http://www.academia.edu/2234136/Programacion_a_gran_escala_generacion_de_columnas)

8. Introducción a la programación matemática. UNED. Universidad complutense de Madrid. J.J.RUZ. (http://www.fdi.ucm.es/profesor/jjruz/MasterUned/Documentos%20en%20aLF/Tema%204.pdf)

9. Problema del transporte y de asignación.
http://davinci.ing.unlp.edu.ar/produccion/catingp/Capitulo%207%20PROBLEMAS%20DE%20TRANSPORTE%20Y%20ASIGNACION.pdf

10. Algoritmo de generación de columnas para la estimación de matrices de viajes Origen-Destino en redes de tráfico congestionadas. D. Verastegui Rayo. (http://www.uclm.es/profesorado/dverastegui/DOCUMENTOS/TESIS/WEBCAP2.pdf).

11. Capítulo 2. Método simplex. (http://catarina.udlap.mx/u_dl_a/tales/documentos/lmnf/diaz_s_s/capitulo2.pdf).

12. Capítulo 3. Método de generación de columnas. (http://catarina.udlap.mx/u_dl_a/tales/documentos/lmnf/diaz_s_s/capitulo3.pdf)