APLICACIÓN DE MÉTODOS DE OPTIMIZACIÓN EN LA SOLUCIÓN DE UN PROBLEMA DE LA INDUSTRIA EDITORIAL

Por:
Cindy V. Calderón Arce.

Tesis sometida en cumplimiento parcial de los requerimientos para el grado de

MAESTRÍA EN CIENCIAS
en
MATEMÁTICA APLICADA

Universidad de Puerto Rico
Recinto Universitario de Mayagüez

Mayo, 2010

Aprobada por:

Daniel McGee, Ph.D. Miembro, Comité Graduado	Fecha
Karen Ríos Soto, Ph.D. Miembro, Comité Graduado	Fecha
Pedro Vásquez Urbano, D.Sc. Presidente, Comité Graduado	Fecha
Mercedes Ferrer Alameda, MSc. Representante de Estudios Graduados	Fecha
Silvestre Colón, MSc. Director del Departamento	Fecha

OPTIMIZATION METHODS APPLIED TO THE SOLUTION OF A PUBLISHING INDUSTRY PROBLEM

By:
Cindy V. Calderón Arce.

Thesis submited in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
APPLIED MATHEMATICS

University of Puerto Rico
Mayagüez Campus

May, 2010

Approved by:

Daniel McGee, Ph.D. Member, Graduate Committee	Date
Karen Ríos Soto, Ph.D. Member, Graduate Committee	Date
Pedro Vásquez Urbano, D.Sc. Chairman, Graduate Committee	Date
Mercedes Ferrer Alameda, MSc. Representative, Graduate Studies	Date
Silvestre Colón, MSc. Department Chair	Date

Resumen de Tesis Presentado a la Escuela Graduada de la
Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

APLICACIÓN DE MÉTODOS DE OPTIMIZACIÓN EN LA SOLUCIÓN DE UN PROBLEMA DE LA INDUSTRIA EDITORIAL

Por:
Cindy V. Calderón Arce.

Mayo, 2010

Consejero: Dr. Pedro Vásquez Urbano.
Departamento: Departamento de Ciencias Matemáticas

RESUMEN

En la industria editorial, se deben crear los negativos de los documentos y las planchas donde se van a colocar dichos negativos para que luego sean enviadas a imprimir. Analizando los costos de imprimir documentos se notó que el costo de crear un negativo es muy elevado en comparación con el costo de imprimir todas las copias necesarias para satisfacer una cierta demanda. Por otro lado, se observó que en muchas ocasiones no se utilizaba todo el espacio disponible en cada una de las planchas o que si todo el espacio era utilizado muchas veces se imprimían copias de más, las cuales representan un desperdicio o un gasto innecesario para la industria.

Como consecuencia de esto, surge el siguiente problema: suponga que una determinada industria debe imprimir n gráficos, cierta cantidad de copias para cada uno de ellos (no necesariamente la misma cantidad para todos), y se tiene una cantidad, no prefijada, de planchas del mismo tamaño, en las cuales se deben colocar primero los negativos de los dibujos para luego enviarlos a imprimir. Lo que se quiere es utilizar la menor cantidad de planchas de tal forma que minimice la cantidad de copias de cada gráfico que no se utilizarán y además el espacio no utilizado en cada plancha.

En el año 2007, se llevó a cabo un proyecto de investigación para resolver el problema de la industria editorial, desarrollando algoritmos basados en heurísticas, cuyos resultados obtenidos dan una solución al problema de una dimensión, únicamente.

Esta investigación tuvo como objetivo principal desarrollar e implementar algoritmos basados en programación lineal que determinen la solución del problema de la industria editorial, utilizando el método de generación de columnas. Se implementaron algoritmos que permiten obtener una solución al problema en una y en dos dimensiones. Con estos algoritmos se lograron mejores resultados que los obtenidos en el 2007, por otro lado, con la solución dada en esta investigación se consigue un menor número de copias sobrantes, lo cual significa disminución en los costos de impresión.

Abstract of Thesis to the Graduate School of the
University of Puerto Rico in partial fulfillment of the
Requirements for the degree of Master of Science

OPTIMIZATION METHODS APPLIED TO THE SOLUTION OF A PUBLISHING INDUSTRY PROBLEM

By:
Cindy V. Calderón Arce.

May, 2010

Chair: Dr. Pedro Vásquez Urbano.
Major Departament: Department Of Mathematical Sciences

ABSTRACT

In the publishing industry negatives have to be created to print documents as well as the plates where the negatives are going to be placed. By weigthing the cost of printing documents it was noted that the cost of creating a negative is higher compared to the cost of printing all the copies required to meet a certain demand. Furthermore, it was observed that all the available space on each plate was not fully used or, if all the space was used more copies were printed, which represent a waste or expense unnecessary for the industry.

As a result, the following problem arises: suppose a particular industry prints n graphics, a certain amount of copies for each (not necessarily the same quantity for each one), and where the size of the plates are all the same.

The negatives of the pictures are prepared first and then sent to print. The goal is to use the least amount of plates in a way that minimizes the number of copies of each graph that are not used and also the unused space on each plate.

A research project was conducted in 2007 to solve the problem of the publishing industry. In that project, algorithms were developed based on heuristics. The results obtained gave a solution only to the one dimensional problem.

Based on the method of column generation this research aims mainly developing and implementing algorithms based on linear programming to determine the solution of the publishing industry problem. Algorithms were implemented that can provide a solution to the problem in one and two dimensions. These algorithms showed better results than those obtain in 2007, also the solution propose in this investigation was able to use less number of waste copies, which is most cost efficient for the publishing industry.

A mi Dios y Señor por darme toda la sabiduría, fortaleza, entendimiento y entusiasmo necesarios para llegar hasta el final.

A toda mi familia por el cariño, el apoyo, la comprensión y la motivación que me han brindado siempre.

Agradecimientos

Primero que todo a Dios por darme esta valiosa oportunidad y permitirme alcanzar una meta más en la vida. A mi familia entera por todo el cariño, todas las palabras de aliento y todo el apoyo dado a la distancia.

A mi consejero Pedro Vásquez por todo el conocimiento transmitido, la comprensión, paciencia, dedicación y disposición brindada. A los profesores Karen Ríos Soto y Daniel McGee por el seguimiento y la ayuda dada. Que sin ellos este trabajo no hubiese sido posible.

A todo el personal del Departamento de Ciencias Matemáticas y a todos mis compañeros de maestría porque de alguna manera formaron parte del proceso. En especial, a los profesores Nilsa Toro y Edgardo Lorenzo por todo su apoyo, las palabras de aliento y la comprensión que me dieron siempre.

Al profesor Luis Acuña por el apoyo, la motivación, por estar siempre dispuesto a ayudarme y porque fue uno de los que siempre estuvo pendiente de mi.

A Mario Marín porque a pesar de la distancia y de lo ocupado de sus días ha estado ahí siempre presente y pendiente de todo, por el cariño, la motivación, la inspiración, la ayuda y el apoyo incondicional.

A César, Milly y Pily por estar ahí siempre presentes, pendientes de mí y de lo que estaba pasando, por todo el apoyo, la ayuda, la comprensión, la motivación, la tolerancia, las palabras de aliento, por hacer que de alguna u otra forma mi experiencia en Puerto Rico sea inolvidable. Sin dejar de lado a María Inés, Raquel y toda su familia, Gabriel, Humberto, Jairo, Laura, Mario, Josue, Lory, Johan, Fila, Mary, Yovani, Pablo por el apoyo, la motivación y la ayuda que me dieron siempre.

A todos gracias infinitas porque de todos aprendí algo, por los buenos y malos momentos, por las risas, los enojos y las angustias... Gracias a todos por todo!!!

Índice general

Lista de Algoritmos

Lista de Tablas

Lista de Figuras

Capítulo 1

Introducción

Esta investigación tiene como objetivo resolver el problema de la industria editorial relacionado con la distribución de los negativos en las planchas, disponibles para generar gráficos.

La optimización busca establecer los valores máximos o mínimos de una determinada circunstancia, proceso o algoritmo. Cualquiera de los casos anteriores puede ser transformado en un problema de programación lineal o no lineal, donde la función objetivo describe la situación que se desea optimizar y las restricciones representan las limitaciones y los requisitos de los cuales depende el problema.

En este trabajo se aplicaron algunos de los resultados más conocidos en el área de optimización, específicamente en programación lineal, con el fin de dar una posible solución al problema de la industria editorial. En general, los problemas de programación lineal tienen múltiples aplicaciones en áreas de la industria y la economía. Entre ellas se pueden mencionar los problemas relacionados con transporte, finanzas, mercadeo, mezclas, logística, producción,

toma de decisiones, inversiones, asignación de tareas, entre otros.

Esta investigación se basó en el método de generación de columnas (*Column Generation*), que tiene sus fundamentos en programación lineal [4, 5, 14, 15]. Se desarrolló un algoritmo que permitió encontrar la solución del problema utilizando métodos y técnicas de programación lineal, redes de flujo (*Network Flows*) y generación de columnas, con el fin de obtener resultados más certeros y confiables. Se implementó el algoritmo utilizando el programa MatLab y se compararon los resultados obtenidos con los presentados en investigaciones anteriores, por medio de pruebas computacionales donde se analizaron los tiempos de ejecución y la solución obtenida en distintas situaciones del mismo problema.

La solución del problema podría traer un gran aporte al área de optimización, principalmente, en problemas relacionados con el problema de la mochila (*Knapsack Problem*) [7], distribución de objetos en recipientes (*Bin Packing Problem*) [12] y cortes (*Cutting Stock Problem*) [14, 6] en los cuales se podría aplicar el método de generación de columnas, ya que el problema de la industria editorial es una variante de los casos mencionados anteriormente. Con eso, investigaciones posteriores pueden utilizar los resultados obtenidos para mejorar la solución de cada uno de ellos, en los cuales muchos investigadores en el área de optimización han trabajado por años.

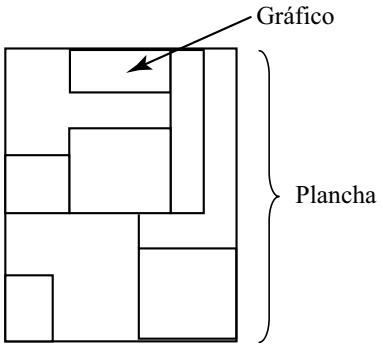
El trabajo está distribuido en seis capítulos, en el presente capítulo se presentó una breve introducción del trabajo realizado y la estructura del mismo. En el Capítulo 2 se describen algunos de los principales trabajos, realizados por otros investigadores, basados en el problema de la industria editorial y en problemas similares a este.

En el Capítulo 3 se presentan los principales fundamentos teóricos en los cuales se basan los resultados presentados. En el Capítulo 4 se describe detalladamente la formulación del problema y el algoritmo solución implementado en MatLab.

En el Capítulo 5 se muestran y se analizan los diferentes resultados obtenidos y se compara el algoritmo implementado con algunos algoritmos desarrollados por otros investigadores. Finalmente, en el Capítulo 6 se resumen las principales conclusiones obtenidas a partir de los resultados presentados en el Capítulo 5 y además se dan algunas recomendaciones para futuras investigaciones relacionadas con el problema de la industria editorial.

Para comprender un poco más los diferentes problemas y situaciones presentadas en el documento, es importante tener claro los siguientes términos:

- Gráfico, ítem, objeto o negativo hace referencia a un documento que se quiere imprimir.
- Cuando se mencione plancha, placa, rollo o barra se hace referencia al molde donde se va a colocar una distribución de negativos de los documentos.
- Un patrón es una distribución de negativos de los documentos que se quieren imprimir.



Capítulo 2

Trabajos previos

En general, no se han podido identificar muchos resultados acerca del problema de la industria editorial. La principal fuente de referencia es una investigación realizada por un estudiante de maestría en el Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional - México, D.F., en la cual se propuso una solución al problema de la industria editorial en una dimensión [2, 11, 12]. En el Capítulo 5 se comparan los resultados obtenidos en esta investigación con los obtenidos en México.

En el año 2007 se llevó a cabo un proyecto de investigación que buscaba resolver el problema de la industria editorial, desarrollando algoritmos basados en heurísticas [2]. A raíz de esto, se publican dos artículos adicionales, en uno de ellos se estableció una relación entre el problema de la industria editorial y el problema de la mochila [11] y en el otro se relacionó el mismo problema con el problema de la distribución de objetos en recipientes [12].

El problema presentado en [12] es el siguiente: Se tienen n gráficos distintos, todos del mismo tamaño, y se desea encontrar m maneras distintas

de colocar de los gráficos en las planchas, de tal forma que se minimice el número de moldes a utilizar siempre y cuando se satisfaga la demanda de cada uno de los gráficos. Matemáticamente, el problema se planteó de la siguiente manera [12]:

$$\begin{aligned} \text{minimizar} \quad & z = \sum_{j=1}^{m} x_j \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} a_{ij} = c;\ j = 1, 2, ..., m \\ & \sum_{i=1}^{m} a_{ij} x_j \geq d_i;\ i = 1, 2, ..., n \\ & x_i \in \mathbb{N} \\ & a_{ij} \in \{1, 2, ..., c\} \end{aligned} \tag{2.1}$$

- c : capacidad de la plancha[1].
- d_i : demanda del i-ésimo objeto[2].
- a_{ij} : cantidad de veces que aparecerá el i-ésimo objeto en la j-ésima plancha.
- x_j : cantidad de copias que se deben sacar de la j-ésima plancha.

El trabajo desarrollado en [11, 12] se describe a continuación en donde se plantea el mismo problema de la industria editorial y se analizan varios casos. Primero, realizan pruebas de la solución dada, utilizando la computadora, con todos los ítems del mismo tamaño, para un solo molde o plancha y luego se

[1]Plancha, molde, barra, placa: base donde se van a colocar los documentos que se quieren imprimir.

[2]Objeto, gráfico, ítem, negativo: documento que se quiere enviar a imprimir

aumenta la complejidad del problema formulándolo de tal forma que haya disponible más de un molde para realizar las impresiones, suponga m moldes.

En el caso de un molde, se supone que el problema tiene las restricciones necesarias para que este se pueda resolver. Primero se coloca un negativo de cada uno de los ítems en dicho molde y se determina la cantidad de copias necesarias para satisfacer todas las demandas con esa "distribución". Si aún queda espacio, se determina cual es el ítem que requiere la mayor cantidad de copias y se agrega un nuevo negativo de ese mismo ítem, se vuelve a determinar la cantidad de copias necesarias. Note que ahora la cantidad de copias necesarias para el ítem que se agregó va a reducirse. Se continúa el proceso hasta que se agote el espacio del molde.

Para el caso de m moldes, lo que se hace es suponer que se tiene un molde más grande de tamaño $m \cdot c$, donde c es el tamaño de cada uno de los moldes pequeños. Entonces, primero se resuelve el problema como el caso anterior para un solo molde de tamaño $m \cdot c$. Después, se llenan uno a uno los moldes, fijando un número máximo de copias (se toma como máximo la demanda del primer objeto), se determina el ítem que requiera el mayor número de copias y se agrega un negativo de ese ítem al molde. Luego, se vuelve a calcular la demanda faltante y así sucesivamente hasta completar la capacidad del molde.

Para el siguiente molde, primero se debe fijar el número máximo de copias, se podría utilizar el mismo máximo anterior pero, como se tienen que satisfacer otras demandas, lo más conveniente sería aplicar de nuevo el proceso anterior, se resuelve el problema para un molde de tamaño $(m-1)c$ (note que después de que se llena el primer molde la demanda de al menos un objeto va

a disminuir). Y el procedimiento se continúa hasta que todas las demandas sean satisfechas.

Por otro lado, se han desarrollado algunas variantes del problema de cortes basados en heurísticas, las cuales utilizan de alguna forma la búsqueda de patrones para encontrar el valor óptimo del problema.

Shiomi et al. [19] presentan un problema de cortes en dos dimensiones en el cual se tienen n ítems, donde el ítem i es de dimensión $w_i \times h_i$ y tiene una demanda d_i $(i = 1, 2, 3..., n)$, se cuenta con una hoja de dimensión $W \times L$, en la cual se van a realizar los cortes. El objetivo es determinar el valor de L de tal forma que se minimice el costo de los cortes y se satisfagan las demandas de todos los ítems, para w_i, h_i, d_i, W y N dados, donde N representa el número máximo de cortes horizontales permitidos y suponiendo que L es mucho más grande que W. Asumiendo que los cortes que se realizan simulan los cortes de una guillotina, el problema se define de la siguiente manera:

$$\begin{aligned} \text{minimizar} \quad & L \\ \text{Sujeto a:} \quad & \text{Número de cortes} \leq N \\ & d_i \leq \text{Número de veces que aparece el ítem } i;\ i = 1, 2, 3, ..., n \end{aligned} \tag{2.2}$$

La solución presentada en [19] para el problema anterior es la siguiente: primero se generan pequeños patrones, distribuciones de los ítems en la hoja, cada uno de esos arreglos va a representar un bloque y el uso iterativo de los bloques es lo que va a representar un arreglo o un patrón como posible solución del problema, suponiendo que el patrón satisface todas las restricciones. Luego, se determina el beneficio o la usabilidad de cada uno de los bloques

antes encontrados y con eso, finalmente, se determina cual es patrón con el cual se alcanza el valor óptimo y con esto el valor de L que satisfaga todas las restricciones.

Otro estudio, es el presentado por Yaodong Cui [17], en el cual dados n piezas rectangulares, cada una de ellas con largo l_i, ancho w_i y utilidad c_i $(i = 1, 2, 3, ..., n)$, y un plato rectangular de largo L y ancho W el objetivo es encontrar el patrón con mayor utilidad, donde la utilidad del patrón es igual a la suma de las utilidades de todas piezas involucradas en dicho patrón. La solución dada en [17] a este problema se basa en métodos recursivos fundamentados en los métodos de programación dinámica y ramificar y acotar (*Branch and Bound*). Primero se elige una pieza y se coloca en el extremo inferior izquierdo del plato. Luego, se realizan dos cortes uno con respecto al ancho y otro con respecto al largo de la pieza elegida, se determina cual de los dos cortes da mayor beneficio y se conserva el que tenga mayor utilidad. El proceso se repite hasta agotar la capacidad del plato o hasta que no sea posible colocar otra pieza en el plato.

Los problemas presentados en [19, 17] son ambos en dos dimensiones. Cui et al. [18] presentan otra variante del problema de cortes pero ahora en una dimensión. El problema presentado en [18] es el siguiente dados n ítems, cada uno de ellos de tamaño l_i y demanda d_i $(i = 1, 2, 3, ..., n)$, y N barras distintas, cada una de ellas de tamaño L_j, costo P_j y tal que de la barra j hayan B_j disponibles $(j = 1, 2, 3, ..., N)$, la solución del problema es una heurística que busca minimizar el costo total y el espacio no utilizado en todos los patrones.

A pesar de que los problemas presentados en [19, 17, 18] están directamente relacionados con el problema de cortes y utilizan la búsqueda y determinación de patrones para su solución, no se puede establecer una relación o comparación entre el problema de la industria editorial y alguno de ellos, por dos razones: una es que el principal objetivo del problema de la industria editorial es la determinación de los patrones de tal forma que minimicen el material desperdiciado y el objetivo de los problemas presentados en [19, 17, 18] está directamente relacionado con los costos y la utilidad de la solución, la segunda es porque los problemas son diferentes y con ellos los datos requeridos son también distintos.

Capítulo 3

Revisión de literatura

Esta investigación está basada en programación lineal y sus aplicaciones. Además, considera resultados importantes relacionados con el problema de la mochila y el problema de cortes. A continuación se hace una descripción de los principales problemas que se utilizan y algunas definiciones básicas que son importantes para obtener los resultados.

3.1. Problema de la mochila

La primera formulación del Problema de la Mochila fue realizada por el ruso Leonid Kantorórich, en 1939. Sin embargo, fue publicada en inglés en el año 1960. Las investigaciones de este problema iniciaron hace 45 años y desde entonces se han presentado diferentes formulaciones o variantes y se han propuesto diversos métodos para resolver dicho problema.

El problema de la mochila, consiste en acomodar en ella diferentes objetos, con un valor asignado que puede medirse de distintas formas: utilidad,

peso, volumen o tamaño. Como es de esperarse la mochila tiene un límite en capacidad ya sea por peso, dimensión o ambas. Lo que se quiere es determinar cuáles objetos se pueden o deben echar en la mochila de tal forma que se obtenga el mayor beneficio de ésta [7].

Por ejemplo, suponga que tiene una fábrica la cual tiene un excedente de $50000, esto le da la posibilidad a la empresa de hacer algún cambio ya sea aumentar la producción, hacer la fábrica más grande, aumentar el personal, diseñar o desarrollar un nuevo producto, algo que le permita aumentar la producción, la oferta o la demanda y con esto el ingreso y la utilidad de dicha empresa. El problema consiste en determinar cuál o cuáles mejoras es más conveniente hacer de tal forma que maximice el beneficio total de la empresa.

En general, el problema de la mochila se puede aplicar a distintas situaciones, tales como restricciones presupuestarias presentes en el área de la economía y la industria, donde lo que se quiere es maximizar el beneficio sin exceder el presupuesto máximo o la determinación de la ruta más corta o más larga, donde minimiza o maximiza, según sea el caso, el costo o beneficio de la ruta.

Matemáticamente: suponga que se tienen n objetos distintos y una mochila. Cada uno de estos objetos tiene un valor $b_i\,(i = 1, 2, ..., n)$ que representa el costo (peso) de colocar el objeto o ítem en la mochila y un valor $c_i\,(i = 1, 2, ..., n)$ que representa la utilidad o importancia (beneficio) del objeto en una determinada situación, además, la mochila tiene una capacidad máxima M. Las variables $x_i\,(i = 1, 2, ..., n)$ representan los objetos que serán analizados. Para cada objeto solo hay dos opciones: se coloca o no en

la mochila, así, los valores que van a tomar las variables x_i son cero o uno, donde el cero indica que el objeto no fue colocado dentro de la mochila y uno que sí. Así, se tiene que el problema de la mochila es un problema binario [7]. La formulación del problema es como sigue:

$$\begin{aligned} \text{maximizar} \quad & z = \sum_{i=1}^{n} c_i x_i \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} b_i x_i \leq M \\ & x_i \in \{0,1\}\,;\, i = 1,2,...,n \end{aligned} \tag{3.1}$$

Similarmente, se puede formular un problema de la mochila para el caso en que se utilice más de una de ellas. Suponga que se tienen n objetos y m mochilas:

$$\begin{aligned} \text{maximizar} \quad & z = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij} \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} b_i x_{ij} \leq M_j;\; j = 1,2,...,m \\ & x_{ij} \in \{0,1\}\,;\, i = 1,2,...,n,\; j = 1,2,...,m \end{aligned} \tag{3.2}$$

- M_j : capacidad máxima de la mochila j.
- b_i : costo del objeto i.
- c_{ij} : utilidad obtenida al colocar el objeto i en la mochila j.
- $x_{ij} = \begin{cases} 0 & ; \text{si no se coloca el objeto } i \text{ en la mochila } j. \\ 1 & ; \text{si se coloca el objeto } i \text{ en la mochila } j. \end{cases}$

Algunos autores consideran como buena estrategia el uso de programación dinámica, para obtener la solución a dicho problema, en ambos casos, cuando se tiene una y cuando se tienen varias mochilas.

3.2. Problema de cortes

Es un problema de programación entera, suponga que se tienen rollos de algún tipo de material, en los cuales se deben hacer cortes de distintos tamaños y cada uno de los tamaños tiene una cierta demanda. Lo que se quiere es determinar la forma de distribuir todos los cortes que se deben realizar de tal manera que se minimice el material desperdiciado.

Según Ehrgott, M., Tind, J. y Teo, C-P se tiene lo siguiente: suponga que cada corte es un ítem u objeto a cortar, primero se formula el problema tal que la función objetivo minimice la cantidad de rollos utilizados, de la siguiente manera:

$$\begin{aligned} \text{minimizar} \quad & z = \sum_j y_j \\ \text{Sujeto a:} \quad & \sum_j x_{ij} = b_i;\ i = 1, 2, 3... \\ & \sum_j w_i x_{ij} = W y_j;\ i = 1, 2, 3... \\ & x_{ij} \in \mathbb{N}^* \\ & y_j \in \{0, 1\} \end{aligned} \tag{3.3}$$

- W : largo de cada rollo.
- $y_j = \begin{cases} 0 & ; \text{si no se utiliza el rollo } j. \\ 1 & ; \text{si se utiliza el rollo } j. \end{cases}$

- w_i : largo o tamaño del objeto i.
- b_i : demanda del objeto i.
- x_{ij} : cantidad de objetos i obtenidos del rollo j.
- $\mathbb{N}^*$: conjunto de los números enteros positivos.

Sin embargo, analizando las ventajas y desventajas de la formulación anterior, se llega a la conclusión de que la matriz de restricciones tiene muchas columnas, lo que convierte al problema en uno difícil de resolver [14]. Por lo que se sugiere el uso de patrones en cada uno de los rollos, un patrón es una posible distribución de los objetos en un rollo. El problema se transforma en uno de minimizar la cantidad de patrones utilizados y con eso la cantidad de rollos cortados (por cada patrón se corta un rollo):

$$\begin{aligned} \text{minimizar} \quad & z = \sum_j x_j \\ \text{Sujeto a:} \quad & \sum_j a_{ij} x_j \geq b_i;\ i = 1, 2, 3... \\ & x_{ij} \in \mathbb{N}^* \end{aligned} \tag{3.4}$$

- x_j : cantidad de rollos utilizados en el patrón j.
- b_i : demanda del objeto i.
- a_{ij} : cantidad de veces que aparece el objeto i en el patrón j.
- $\mathbb{N}^*$: conjunto de los números enteros positivos.

A pesar de la transformación que se hizo, el problema sigue teniendo una desventaja y es debido a la cantidad de patrones posibles que se pueden formar, por lo que para empezar se toman solo unos cuantos de esos patrones. Generalmente, y a conveniencia, se inicia el problema con aquellos patrones que formen la matriz identidad en la matriz de restricciones. Es decir, los patrones en los cuales de cada rollo se corte solo un objeto, sin embargo, cualquier otro patrón podría ser tomado en cuenta para iniciar. Por la forma del problema, en el cual no se van a conocer todas las columnas de la matriz de restricciones el método utilizado para encontrar la solución es el de generación de columnas [5, 14, 15].

3.3. Generación de columnas

Es un algoritmo que se utiliza en problemas con muchas variables o bien cuando no se conocen todas las columnas de la matriz de restricciones. El uso de muchas variables nos lleva a una matriz de restricciones con muchas columnas, lo cual podría convertir un problema "sencillo" en uno complicado y difícil de resolver [4, 5].

Considere el siguiente problema de cortes, tal y como se definió anteriormente:

$$\begin{aligned} \text{minimizar} \quad & z = \sum_j x_j \\ \text{Sujeto a:} \quad & \sum_j a_{ij} x_j \geq b_i; \ i = 1, 2, 3... \\ & x_j \in \mathbb{N}^* \end{aligned} \tag{3.5}$$

El método de generación de columnas empieza con algunas columnas y a medida que se avanza, se van agregando aquellas columnas que ayuden a mejorar la solución del problema o por el contrario se eliminan aquellas que en realidad no dan gran aporte a la solución. Así, el Problema Maestro (*Master Problem*), problema principal que minimiza el desperdicio, está dado por:

$$\begin{aligned} \text{minimizar} \quad & z = \sum_k x_{j\,k} \\ \text{Sujeto a:} \quad & \sum_k a_{ij_k} x_{j_k} \geq b_i;\ i = 1, 2, 3... \\ & x_{j_k} \in \mathbb{N}^* \end{aligned} \tag{3.6}$$

Luego se empiezan a agregar las columnas que sean necesarias, por medio del siguiente proceso:

1. Generar posibles patrones del problema y formular el primer Problema Maestro con esas columnas (patrones).

2. Resolver el Problema Maestro anterior relajado, es decir, se resuelve el problema para el caso en el cual las variables se consideran continuas y no enteras.

3. Con base en la solución del problema relajado, determinar una solución entera del problema dado.

4. Calcular las variables duales del problema previo.

5. Obtener un Subproblema (*Pricing Problem*), derivado del Problema Maestro:

$$\begin{aligned} \text{maximizar} \quad & z = \sum_i \pi_i a_{ij} \\ \text{Sujeto a:} \quad & \sum_i l_i a_{ij} \leq L \\ & a_{ij} \geq 0 \end{aligned} \tag{3.7}$$

- π_i: variables duales del Problema Maestro anterior.
- l_i: largo de cada uno de los ítems.
- L: largo de cada rollo.
- j: es fijo, representa la columna (patrón) que se está analizando.
- a_{ij}: variables, que en este caso representan la nueva columna obtenida.

El cual minimiza el desperdicio o espacio que no se utiliza en cada uno de los rollos o, lo que es equivalente, maximiza el beneficio de cada rollo. Y con eso se determina cual columna es más conveniente agregar al sistema original (Problema Maestro).

6. Agregar la columna obtenida (A_j) al Problema Maestro formulado al inicio y obtener un nuevo Problema Maestro.

7. Repetir el proceso hasta que ya no se obtengan más columnas factibles, es decir, hasta que el valor de la función objetivo del Subproblema no sea factible. Eso significa, que ninguna de las columnas fuera del Problema Maestro va a contribuir a disminuir, aún más, el valor de la función objetivo del problema general.

3.3.1. Métodos auxiliares

Descomposición de Dantzig-Wolfe (*Dantzig-Wolfe Decomposition*):

Se utiliza para descomponer el Problema Maestro en Subproblemas un poco más pequeños. Este método ayuda a resolver problemas de programación lineal con ciertas condiciones sobre la matriz de restricciones. Primero, se deben identificar y agrupar aquellas filas en las que la mayoría de los coeficientes de las variables involucradas sean diferentes de cero. Luego, debe ser posible descomponer las filas restantes en submatrices, de tal forma que si una variable tiene coeficiente distinto de cero en una submatriz en las demás matrices dicho coeficiente debe ser igual a cero [8].

Método Simplex (*Simplex Method*)

Resuelve el Problema Maestro y cada uno de los Subproblemas. Además, ayuda a determinar el costo mínimo de las columnas que no han sido tomadas en cuenta y, con ello, establecer la columna que se debe agregar.

Dado que la región factible de un problema de programación lineal es un poliedro, el método Simplex consiste en ir buscando la solución óptima del problema partiendo de cierto vértice y moviéndose a través de la frontera, donde cada iteración es una prueba o análisis en los puntos extremos de dicha región. El proceso continúa hasta que el valor de la función objetivo no pueda ser mejorado [9].

En general, el método de simplex es un proceso iterativo que se basa en el método de Gauss-Jordan donde el pivote es la intersección entra la fila de la variable saliente y la columna de la variable entrante.

Considere el siguiente problema de programación lineal, en posición estándar:

$$\begin{aligned} \text{minimizar} \quad & z = c^T x \\ \text{Sujeto a:} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.8}$$

Así, se propone para el método Simplex el siguiente proceso:

1. Determinar el conjunto de índices de las variables básicas (I) y el conjunto de índices de las variables no básicas (J). Así, el problema 3.8 se puede escribir como:

$$\begin{aligned} \text{minimizar} \quad & z = \left(c_B^T : c_N^T\right) \begin{pmatrix} x_B \\ x_N \end{pmatrix} \\ \text{Sujeto a:} \quad & (B : N) \begin{pmatrix} x_B \\ x_N \end{pmatrix} = b \\ & x_B \geq 0 \\ & x_N = 0 \end{aligned} \tag{3.9}$$

   - b: vector constante.
   - x_B: variables básicas.
   - x_N: variables no básicas.
   - c_B: coeficientes de las variables básicas.
   - c_N: coeficientes de las variables no básicas.
   - B: columnas de la matriz A que corresponden a las variables básicas.
   - N: columnas de la matriz A que corresponden a las variables no básicas.

2. Determinar $c_N^T - c_B^T B^{-1} N$.

   - Si $c_N^T - c_B^T B^{-1} N \geq 0$, entonces la solución es óptima.
   - Si $c_N^T - c_B^T B^{-1} N < 0$, entonces la solución se puede mejorar.

3. Determinar la variable entrante x_t:

$$t = \min_{j \in J} \left\{ c_j - c_B^T B^T N_j < 0 : N_j \text{ es la } j\text{-ésima columna de } N \right\}$$

4. Determinar la variable saliente x_s:

$$s = \min_{i \in I} \left\{ \frac{(B^{-1}b)_i}{(B^{-1}N_t)_i} : (B^{-1}N_t)_i > 0 \right\}$$

5. Actualizar:

   $I : (I - \{x_s\}) \cup \{x_t\}$

   $J : (J - \{x_t\}) \cup \{x_s\}$

   B : la columna de la s-ésima variable del B anterior se sustituye por la columna de la t-ésima variable del N anterior.

   N : la columna de la t-ésima variable del N anterior se sustituye por la columna de la s-ésima variable del B anterior

   c_B : el elemento s del c_B anterior se sustituye por el elemento t del c_N anterior

   c_N : el elemento t del c_N anterior se sustituye por el elemento s del c_B anterior

   $x_B = B^{-1}b$

   $x_N = 0$

6. Repetir los pasos 2, 3, 4 y 5 hasta alcanzar el óptimo.

Holgura complementaria (*Complementary slackness*)

Usado para determinar los coeficientes o costos de la función objetivo en cada Subproblema. Este resultado ayuda a determinar los valores de las variables duales o multiplicadores de Lagrange, sin necesidad de resolver el dual de un problema de programación lineal.

Teorema: "Sean x y y las soluciones del siguiente par de problemas, primal y dual respectivamente,

$$\begin{array}{rl|rl} \text{minimizar} & z = c^T x & \text{maximizar} & w = b^T y \\ \text{Sujeto a:} & Ax \geq b & \text{Sujeto a:} & A^T y \leq c \\ & x \geq 0 & & y \geq 0 \end{array} \tag{3.10}$$

entonces se cumplen las siguientes proposiciones:

i. $x_i > 0 \Rightarrow y^T A_i = c_i$, donde A_i representa la columna i de la matriz A.

ii. $y^T A_i < c_i \Rightarrow x_i = 0$, donde A_i representa la columna i de la matriz A.

iii. $y_j > 0 \Rightarrow A^j x = b_j$, donde A^j representa la fila j de la matriz A.

iv. $A^j x > b_j \Rightarrow y_j = 0$, donde A_j representa la fila j de la matriz A"[9].

Según el teorema anterior se tiene que dadas las soluciones del primal (Problema Maestro) x_i, si estas son mayores que cero, entonces, las soluciones duales (Subproblema) y_i cumplen la siguiente relación:

$$y^T a_{ij} = c_j$$

Si alguna solución del primal x_i es exactamente cero, entonces únicamente se puede asegurar que:

$$y^T a_{ij} \leq c_j$$

3.4. Problemas de programación lineal entera

Son aquellos problemas de programación lineal donde se requiere que las soluciones sean enteras, tal y como se indica en el siguiente problema:

$$\begin{aligned} \text{minimizar} \quad & z = c^T x \\ \text{Sujeto a:} \quad & Ax \geq b \\ & x \in \mathbb{N} \end{aligned} \tag{3.11}$$

Para determinar la solución a ese tipo de problemas se tienen los siguientes métodos:

- Ramificar y acotar (*Branch and bound*)

  Este algoritmo resuelve primero el problema relajado, es decir, sin la restricción de que las soluciones sean enteras y a partir de los datos obtenidos hace cortes de planos y vuelve a resolver el problema con el corte que se hizo. Si la solución de ese segundo problema es mejor que la obtenida anteriormente, se sigue analizando el problema con ese corte y se "desecha" la parte que esta fuera de él. En caso contrario, se "desecha" el corte que se acaba de realizar y se continua con la parte que esta fuera del mismo. El proceso sigue hasta que la nueva solución entera obtenida no mejore la solución anterior. La idea de los cortes es irse aproximando cada vez más a una solución entera del problema.

- Cortes de planos (*Cutting plane*)

  Este método ayuda al algoritmo anterior. Realiza cortes de planos sobre la región factible del problema para así disminuir el espacio y, con esto, los casos o posibles soluciones que deben analizarse.

3.5. Problemas tipo NP-Difícil (*NP-Hard problems*)

Un problema es de la clase NP si y sólo si se puede resolver por un algoritmo no determinístico con complejidad de tiempo polinómico, en otras palabras si y sólo si el problema se puede resolver por un algoritmo polinomial no determinístico. Es importante notar que polinomial no determinístico no es lo mismo que no polinomial [13].

Se dice que un problema es de la clase NP-Completo (*NP-Complete*) si el problema es computacionalmente tan difícil como cualquier problema razonable sujeto a una formulación precisa, son problemas de la clase NP donde su resolución involucra una alta dificultad computacional [10]. Este tipo de problemas satisfacen las siguientes propiedades:

1. Ningún problema NP-Completo puede ser resuelto por un algoritmo polinomial conocido.

2. Si existiera algún algoritmo polinomial para resolver un problema NP-Completo, entonces hay algoritmos polinomiales para resolver todos los problemas NP-Completos.

Se cree que puede haber algoritmos no polinomiales para resolver problemas NP-Completos, sin embargo, eso no ha podido ser demostrado [10].

Por otro lado, en muchos casos se puede probar que algunos problemas de la clase NP pueden ser reducidos a un problema polinomial. Sin embargo, no se puede afirmar que dicho problema es de la clase NP-Completo a pesar de que posiblemente el problema sea muy difícil de resolver y con esto probablemente intratable. A ese tipo de problemas se les llama NP-Difícil

(*NP-Hard*) [10]. Uno de los problemas *NP-Hard* más conocido es el problema de la mochila (*Knapsack Problem*) [13].

Capítulo 4

Formulación del problema de la industria editorial

El problema de la industria editorial se caracteriza por tener situaciones de impresión las cuales implican una inversión en la reproducción de las copias que en algunas ocasiones generan costos exagerados. Esta investigación se enfoca en formular, como un problema de optimización, la producción de negativos de los gráficos que se utilizan en la industria editorial. El objetivo es minimizar los costos de la compra de las planchas de papel que se utilizan para generar los negativos de los gráficos o equivalentemente, minimizar la cantidad de gráficos en exceso que se producen.

Este es un problema clásico de programación lineal entera y se puede formular como un problema en una dimensión o en dos dimensiones. Nuestro objetivo es encontrar soluciones óptimas razonables en el menor tiempo posible.

4.1. Problema en una dimensión

En este problema se tiene un conjunto de n gráficos de tamaño $l_1, l_2, l_3, ..., l_n$, donde cada uno de ellos cuenta con una demanda $d_1, d_2, d_3, ..., d_n$, respectivamente. Las planchas a utilizar son todas de tamaño L. El objetivo es minimizar la cantidad de patrones que se generen y la producción de gráficos en exceso. El problema se visualiza de forma tal que las planchas y los gráficos son segmentos, como lo ilustra la Figura 4.1:

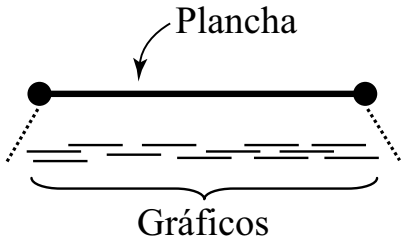


Figura 4.1: Problema de la industria editorial en una dimensión.

4.1.1. Formulación del problema en una dimensión

Este problema se caracteriza por que considera la siguiente información:

- d_i : demanda del i-ésimo gráfico.
- c_{ij} : cantidad de veces que aparece el gráfico i en el patrón j.
- x_j : cantidad de copias que se deben sacar del j-ésimo patrón.

El problema en una dimensión se puede formular como un problema de programación lineal entera de la siguiente manera:

$$\begin{aligned} \text{minimizar} \quad & \sum_{j}^{m} x_j \\ \text{Sujeto a:} \quad & \sum_{j}^{m} c_{ij} x_j \geq d_i;\ i = 1, 2, 3..., n \\ & x_j \in \mathbb{N}^*;\ j = 1, 2, 3, ..., m \end{aligned} \tag{4.1}$$

donde:

- $\sum_{j}^{m} x_j$ es la función objetivo que representa el total de copias que se deben sacar de todos los patrones (planchas).
- $\sum_{j}^{m} c_{ij} x_j \geq d_i$ es la restricción que garantiza que todas las demandas sean cumplidas.
- m no es una cantidad prefijada.

A este problema se le llamara *Problema Maestro* y es un problema *NP-Hard* debido a que cada Subproblema es un pequeño problema de la mochila y además el problema completo involucra un gran número de variables. Es por esa razón que el problema se resuelve como un problema relajado de programación lineal con variables continuas no negativas y luego se establece una aproximación entera de las soluciones que se obtienen.

Como m puede llegar a ser muy grande, puede ser muy difícil resolver el problema de programación entera, esto nos lleva a la idea de utilizar generación de columnas. En este caso, se espera que el número de iteraciones

dependa del número de gráficos n y no de la cantidad de patrones dados m. Para resolver el problema lineal usando generación de columnas, se debe cumplir que:

i. Se tiene una base inicial.

ii. Determinar si las columnas no básicas tienen costo reducido mayor o igual que cero y si no, encontrar una columna con costo reducido negativo.

La base inicial se puede conseguir considerando los elementos:

$$a_{ij} = \begin{cases} 1 & ; \text{ si } i = j \\ 0 & ; \text{ si } i \neq j \end{cases}$$

Así:

$$x_j \begin{cases} \neq 0 & ; \text{ si } j = 1, 2, ..., m \\ = 0 & ; \text{ en otro caso} \end{cases}$$

Luego considere el proceso de generar nuevos patrones, esto es, encontrar una columna no básica cuyo costo reducido sea negativo. Dada una solución factible, la solución dual correspondiente se puede obtener utilizando holgura complementaria. Resolviendo para un determinado patrón, de la siguiente manera $c_j - yC_j = 1 - yC_j$, donde C_j es la j-ésima columna de la matriz de

coeficientes del *Problema Maestro* previo. Lo discutido anteriormente permite obtener el siguiente problema de programación lineal:

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} y_i c_{ij} \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} l_i c_{ij} \leq L \\ & c_{ij} \in \mathbb{N} \end{aligned} \tag{4.2}$$

- y : variables duales del Problema Maestro anterior.
- l_i : largo del i-ésimo gráfico.
- L : largo del molde.
- $c_{:j}$: nuevo patrón.

Así, el proceso general para resolver el problema completo está dado de la siguiente manera; primero se generan posibles patrones, para iniciar el método de generación de columnas. En este caso, la matriz de los primeros patrones será la matriz identidad de orden n, es decir, se coloca un gráfico por molde. Luego, se resuelve el primer *Problema Maestro* con la primer matriz de patrones (identidad). Note que la solución de este primer problema es inmediata, así el vector x (solución) es igual al vector d (demanda). Y en este caso no es necesario utilizar el algoritmo `AproxMast` (Ver apéndice).

Después, se determinan las variables duales del problema anterior para poder formar el primer Subproblema. En este primer caso $x = d$ por ende, todas las variables duales van a ser igual a uno. Luego, se resuelve el primer

Subproblema utilizando el algoritmo `AproxPric` (Ver apéndice). La solución de dicho Subproblema representa un nuevo posible patrón.

Si el nuevo patrón mejora la solución del problema general, se añade a la matriz de patrones. Se continúa buscando otro posible patrón por medio del siguiente procedimiento:

1. Se resuelve el nuevo *Problema Maestro*, con la nueva matriz de patrones, utilizando el algoritmo `AproxMast` (Ver apéndice).

2. Si alguna de las soluciones del Problema Maestro anterior es cero, indica que el patrón asociado a dicha variable no se está utilizando por lo que no es necesario que este dentro de la matriz de patrones y en ese caso se elimina de ella.

3. Se determinan las variables duales asociadas al *Problema Maestro* previo.

4. Se resuelve el nuevo Subproblema, utilizando el algoritmo `AproxPric` (Ver apéndice).

El proceso anterior se repite hasta que se encuentre un patrón que no mejor la solución del problema general. En ese caso ya se han encontrado todos los posibles patrones que optimizarán el problema, así se obtiene T (Ver Algoritmo 4.1).

Finalmente, se resuelve el último Problema Maestro para determinar la solución óptima x.

El proceso anterior se resume a continuación:

Algoritmo 4.1: Aproximación al problema en una dimensión

`[T,x]=UnaDimen(d,l,L)`

Entrada: $d \in \mathbb{R}^n$; $l \in \mathbb{R}^n$; $L \in \mathbb{R}$

Salida: $T \in \mathbb{R}^{n \times p}$; $x \in \mathbb{R}^p$

1 $P \leftarrow$ posibles patrones iniciales

2 **while** *No se hayan encontrado todos los patrones convenientes* **do**

3 Resolver el Problema Maestro

4 $y \leftarrow$ Variables duales del Problema Maestro

5 Resolver el Subproblema

6 **end**

7 Resolver el último Problema Maestro (x)

4.2. Problema en dos dimensiones

En este problema se tiene un conjunto de n gráficos, donde el gráfico i tiene ancho a_i, largo b_i y demanda d_i $(i = 1, 2, 3, ..., n)$ y planchas de ancho A y largo B. El objetivo es minimizar la cantidad de patrones que se generen y la producción de gráficos en exceso. El problema se visualiza de forma tal que las planchas y los gráficos son rectángulos, como se muestra en la Figura 4.2:

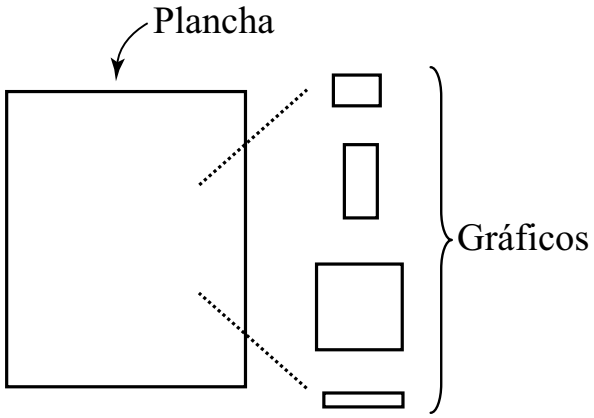


Figura 4.2: Problema de la industria editorial en dos dimensiones.

4.2.1. Formulación del problema en dos dimensiones

Este problema se caracteriza por que considera la siguiente información:

- d_i : demanda del i-ésimo gráfico.
- c_{ij} : cantidad de veces que aparece el gráfico i en el patrón j.
- x_j : cantidad de copias que se deben sacar del j-ésimo patrón.

El problema en dos dimensiones es un problema de optimización combinatorial y se puede formular, de forma similar al problema en una dimensión, como un problema de programación lineal entera:

$$\begin{aligned} \text{minimizar} \quad & \sum_j x_j \\ \text{Sujeto a:} \quad & \sum_j c_{ij} x_j \geq d_i;\ i = 1, 2, 3..., n \\ & x_j \in \mathbb{N}^* \end{aligned} \tag{4.3}$$

El problema 4.3 se genera de la misma manera que el problema 4.1 y de esa forma la solución de ambos problemas es la misma. Por esa razón, el problema 4.3 también representa un problema de la clase NP-Hard.

Para determinar la nueva posible columna o el nuevo posible patrón se formulan dos Subproblemas, uno con respecto al largo y otro con respecto al ancho, definidos de la siguiente manera:

I. Subproblema con respecto al largo:

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} y_i u_{ij} \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} b_i u_{ij} \leq B \\ & u_{ij} \in \mathbb{N} \end{aligned} \tag{4.4}$$

- y : variables duales del *Problema Maestro* anterior.
- b_i : largo del i-ésimo gráfico.
- B : largo del molde.
- $u_{:j}$: patrón respecto al largo.

II. Subproblema con respecto al ancho:

$$\begin{aligned} \text{maximizar} \quad & \sum_k y_{i_k} v_{i_k j} \\ \text{Sujeto a:} \quad & \sum_k a_{i_k} v_{i_k j} \leq A \\ & v_{i_k j} \in \mathbb{N} \end{aligned} \tag{4.5}$$

- y_{i_k} : variable dual del Problema Maestro anterior, correspondiente al gráfico i_k.
- a_{i_k} : ancho del gráfico i_k.
- A : ancho del molde.
- $v_{:j}$: patrón respecto al ancho para el k-ésimo elemento del patrón anterior, con respecto al largo.
- Los gráficos i_k se eligen de tal forma que cumpla que $b_{i_k} \leq b_k$.

Similarmente al problema en una dimensión, primero se generan posibles patrones, para iniciar el método de generación de columnas, donde la matriz de los primeros patrones será la matriz identidad de orden n (n: cantidad de gráficos). Luego, se resuelve el primer *Problema Maestro* con la primer matriz de patrones (identidad). Para este primer *Problema Maestro*, igualmente al caso anterior, $x = d$. Seguidamente, se determinan las variables duales del primer *Problema Maestro*, las cuales van a ser todas iguales a uno.

Para determinar el nuevo posible patrón, se toman en cuenta dos Subproblemas: uno con respecto al largo y otro con respecto al ancho, cada uno de esos Subproblemas en una dimensión. Los Subproblemas se resuelven por medio del algoritmo `AproxMast` (Ver apéndice) y la combinación de ambas soluciones representa el nuevo patrón. (Ver siguiente subsección)

Si el nuevo patrón mejora la solución del problema general, se añade a la matriz de patrones. Y se continúa buscando otro posible patrón de la siguiente manera:

1. Se resuelve el nuevo *Problema Maestro*, con la nueva matriz de patrones, utilizando el algoritmo `AproxMast` (Ver apéndice).

2. Si alguna de las soluciones del *Problema Maestro* anterior es cero, indica que el patrón asociado a dicha variable no se está utilizando por lo que no es necesario que este dentro de la matriz de patrones y en ese caso se elimina de ella.

3. Se determinan las variables duales asociadas al *Problema Maestro* previo.

4. Se determina un nuevo posible patrón, por medio de las soluciones de los dos Subproblemas.

El proceso anterior se repite hasta que se encuentre un patrón que no mejore la solución del problema general. En ese caso ya se ha encontrado la matriz T de todos los posibles patrones que optimizan la solución del problema.

Finalmente, se resuelve el último *Problema Maestro* para determinar la solución óptima x.

En general, se tiene:

Algoritmo 4.2: Aproximación al problema en dos dimensiones

`[T,x]=DosDimen(a,b,d,A,B)`

Entrada: $a \in \mathbb{R}^n$; $b \in \mathbb{R}^n$; $d \in \mathbb{R}^n$; $A \in \mathbb{R}$; $B \in \mathbb{R}$

Salida: $T \in \mathbb{R}^{n \times p}$; $x \in \mathbb{R}^p$

1 $P \leftarrow$ posibles patrones iniciales
2 **while** *No se hayan encontrado todos los patrones convenientes* **do**
3 Resolver el Problema Maestro
4 $y \leftarrow$ Variables duales del Problema Maestro
5 Resolver el Subproblema con respecto al largo
6 Resolver los Subproblemas con respecto al ancho
7 Crear el nuevo patrón
8 **end**
9 Resolver el último Problema Maestro (x)

¿Cómo crear el nuevo posible patrón?

Se definen dos Subproblemas, en una dimensión cada uno de ellos, uno para el largo y otro para el ancho. Primero se define el Subproblema con respecto al largo:

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} y_i u_{ij} \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} b_i u_{ij} \leq B \\ & u_{ij} \in \mathbb{N} \end{aligned}$$

y se resuelve por medio del algoritmo `AproxPric` (Ver apéndice). La solución obtenida es un "patrón" con respecto al largo.

Una vez resuelto el problema anterior, se organizan los gráficos en "líneas", cada una de las líneas va a estar definida por los gráficos que aparecen en la solución obtenida anteriormente, es decir, por cada $u_{ij} \neq 0$ va a haber una línea.

Suponga que $u_{rj} \neq 0$, en la línea correspondiente al gráfico r van a estar todos aquellos gráficos que con largo menor o igual al largo de r. Es decir, si $b_s \leq b_r$ entonces el gráfico s está en la línea que corresponde al gráfico r.

Pueden ocurrir dos cosas:

1. En una línea hay sólo un gráfico.

2. Un mismo gráfico puede estar en más de una línea.

Por cada línea existe un Subproblema, con respecto al ancho, de la siguiente forma:

$$\begin{aligned} \text{maximizar} \quad & \sum_k y_{i_k} v_{i_k j} \\ \text{Sujeto a:} \quad & \sum_k a_{i_k} v_{i_k j} \leq A \\ & v_{i_k j} \in \mathbb{N} \end{aligned}$$

Finalmente, la $i-$ésima entrada del patrón j es igual a la suma de la cantidad de veces que aparece el $i-$ésimo gráfico en cada una de las soluciones de los problemas con respecto al ancho.

Capítulo 5

Discusión de resultados

Para analizar y comparar los resultados y la efectividad de los métodos propuestos se realizaron pruebas en problemas con diferentes cantidades de gráficos a imprimir. La cantidad de gráficos en cada uno de los problemas varía desde 10 hasta 5000, las dimensiones de cada uno de ellos están en el rango $[20, 40]$ y las demandas en $[25000, 50000]$, ambos datos son generados por medio de un generador de números aleatorios basado en una distribución discreta uniforme . Lo anterior, con el objetivo de que cada uno de los problemas asemeje problemas reales y además para que los datos de los diferentes problemas generados sean más uniformes. Es decir, que la variación en las dimensiones de los gráficos y las demandas sea pequeña y no haya gran diferencia entre un problema y otro, aparte de la cantidad de gráficos que son considerados.

Cada uno de los problemas es generado de forma aleatoria e independiente. Ya que por ejemplo suponiendo que se ha generado un problema con 10 gráficos y que ahora se quiere generar uno con 20 gráficos, da lo mismo si

se mantienen los 10 datos anteriores y se agregan 10 nuevos o si se generan todos los 20 nuevamente, debido a que no se puede garantizar nada acerca de la variación de los 10 nuevos datos.

Además, en cada uno de los algoritmos, se fijó un límite máximo de 100000 iteraciones para cada problema que se quiera resolver.

5.1. Ejemplos

5.1.1. Una dimensión

Se tienen planchas, en una dimensión, todas de tamaño 100 y se deben enviar a imprimir 10 gráficos con los datos presentados en la Tabla 5.1.

n	1	2	3	4	5	6	7	8	9	10
l_n	29	29	35	23	25	22	31	31	26	21
d_n	40614	46497	26335	25040	26221	34350	29336	30855	40858	33780

Tabla 5.1: Problema en una dimensión, con 10 ítems y moldes de capacidad 100.

donde l_n representa el tamaño y d_n la demanda del n-ésimo gráfico. Se quiere buscar la forma óptima de acomodar los negativos de los gráficos en las planchas de tal forma que se minimice el exceso de copias de cada gráfico y el espacio no utilizado en cada plancha.

Al aplicar el algoritmo `UnaDimen` (página 31) se obtiene la siguiente solución:

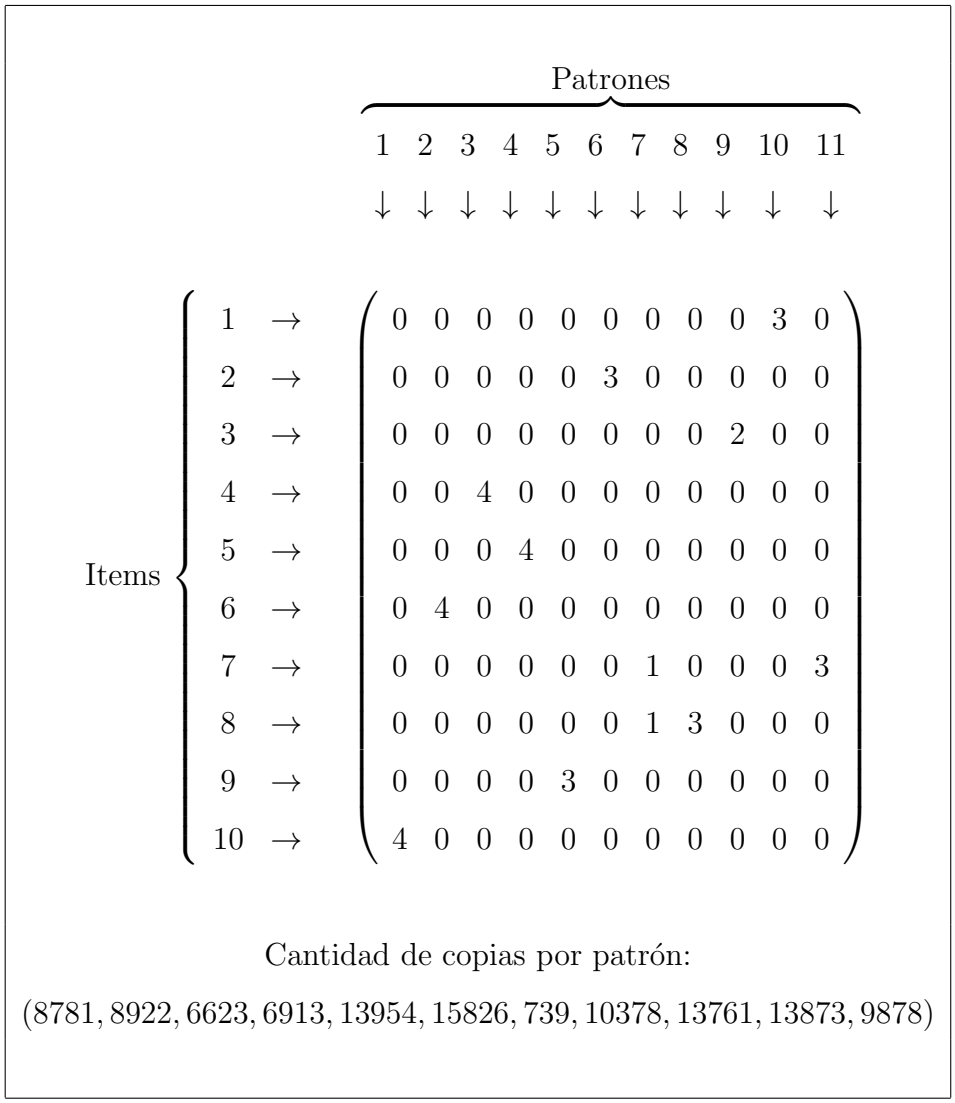


Figura 5.1: Solución del problema presentado en la Tabla 5.1.

Los datos de la Figura 5.1 indican, por ejemplo, que el gráfico 8 aparece en el patrón 7 una vez y en el patrón 8 tres veces. Además, note que en el

séptimo patrón también está el gráfico 7 una vez. Con lo que se tiene que en el molde del patrón 7 se están utilizando 62 unidades y en el molde del patrón 8 se utilizan 93 unidades, de un total de 100 para cada molde (Ver Tabla 5.1).

Así mismo, del patrón 7 se deben imprimir 739 copias y del 8 se deben imprimir 10378 copias, en ese caso se obtendrán $739 + 3 \cdot 10378 = 31873$ copias del octavo gráfico. Siguiendo la Tabla 5.1, la demanda de ese gráfico es únicamente 30855 copias, por lo que se imprimirán 1018 copias de más del gráfico 8.

5.1.2. Dos dimensiones

Se tiene planchas, en dos dimensiones, todas de tamaño 100×100 y se deben imprimir 10 gráficos con los datos presentados en la Tabla 5.2.

n	1	2	3	4	5	6	7	8	9	10
a_n	32	35	22	23	38	28	28	20	24	20
b_n	33	36	40	20	35	25	40	22	20	33
d_n	45043	28734	42259	36933	29300	36488	33321	44538	31298	33048

Tabla 5.2: Problema en dos dimensiones, con 10 ítems y moldes de capacidad 100×100.

donde a_a representa el ancho, b_n el largo y d_n la demanda del n-ésimo gráfico. Se quiere determinar el acomodo óptimo de los negativos de los gráficos en las planchas de tal forma que se minimice la cantidad de copias sobrantes, de cada gráfico, y el espacio no utilizado en cada plancha.

Al aplicar el algoritmo `DosDimen` (página 36) se obtiene la siguiente solución:

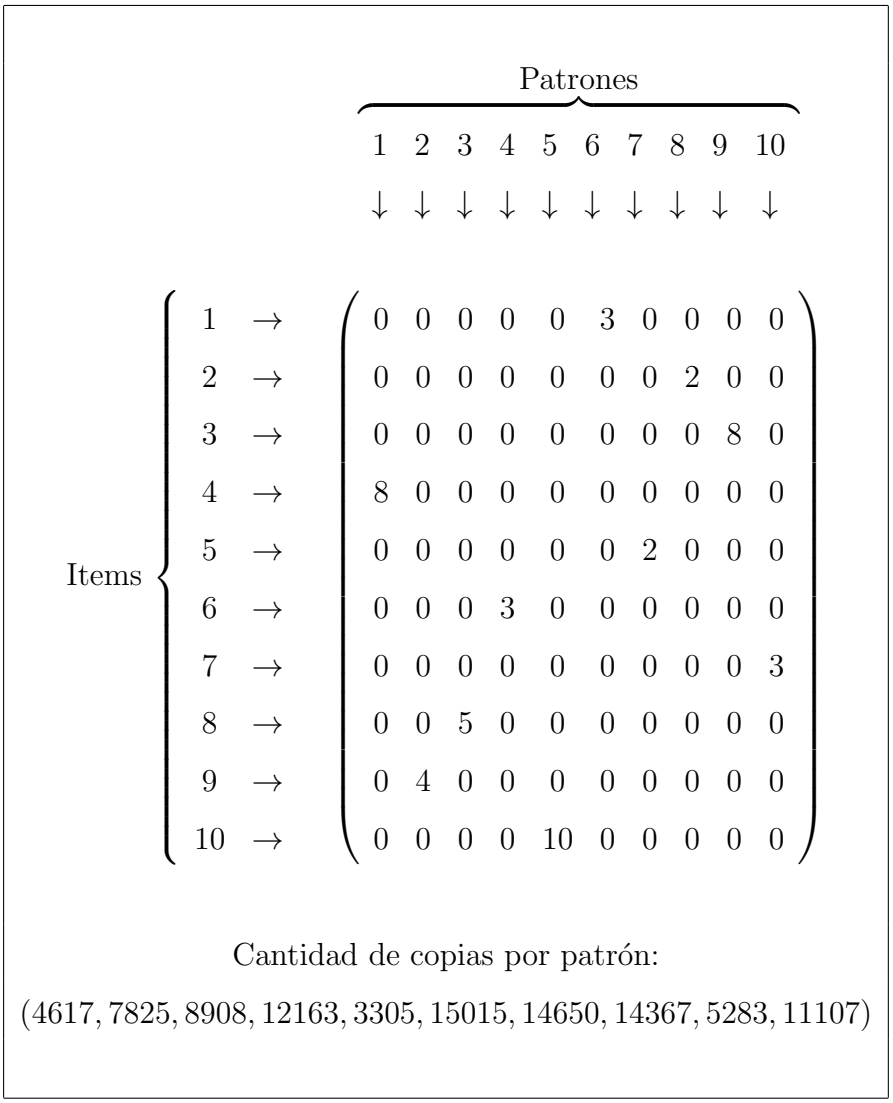


Figura 5.2: Solución del problema presentado en la Tabla 5.2.

Los datos de la Figura 5.2 representan, por ejemplo, que el gráfico 8 aparece en el patrón 3 cinco veces. Con lo que se tiene que en el molde del

patrón 3 se están utilizando 2200 unidades cuadradas, de un total de 100000 por cada molde (Ver Tabla 5.2).

Así mismo, del patrón 3 se deben imprimir 8908 copias, en ese caso se obtendrán 44540 copias del octavo gráfico. Siguiendo la Tabla 5.2, de ese gráfico se necesitaban 44538 copias, por lo que se imprimirán en exceso únicamente 2 copias del gráfico 8.

Note que cada una de las columnas de la matriz de patrones representa una plancha o patrón y los datos de la esa columna muestran cómo van a estar distribuidos los negativos de cada uno de los gráficos en esa plancha, en otras palabras, la cantidad de negativos de cada gráfico que se deben colocar en dicha plancha.

5.2. Comparación de resultados

La comparación de los resultados obtenidos en esta investigación con los obtenidos en [2, 11, 12] se realizará únicamente con problemas en una dimensión. Pues para el caso de dos dimensiones no hay criterio de comparación, pues en la revisión de la literatura no se identificaron problemas de optimización en dos dimensiones similares al desarrollado en esta investigación.

Además, con el fin de justificar la heurística implementada para aproximar la solución entera del *Problema Maestro* y del Subproblema (Ver apéndice), se realizan comparaciones entre los resultados obtenidos utilizando dicha heurística con los resultados obtenidos utilizando el programa `lp_solve`[1].

[1]Tomado de http://sourceforge.net/projects/lpsolve/files/lpsolve/
Versión lp_solve_5.5.0.15_MATLAB_exe_win32.zip

Este programa está basado en el método ramificar y acotar (*Branch and bound*), el cual determina la solución de un problema de programación lineal entero. En ambos casos los algoritmos utilizados son los mismos (`UnaDim`, `DosDim`), lo único que varía de uno a otro es la forma de resolver los problemas de programación lineal entera (Calderón-Vásquez-Heurística, Calderón-Vásquez-`lp_solve`).

5.2.1. Una dimensión

Considere el siguiente problema de la industria editorial en una dimensión, presentado por Sagols F. et al. en [12]: Se tienen planchas, en una dimensión de tamaño 12 y se desean imprimir copias de seis gráficos, tal y como se presenta a continuación:

- $n = 6$
- $L = 12$
- $l = (1, 1, 1, 1, 1, 1)$
- $d = (4736, 2464, 7082, 14226, 9014, 14032)$

En la siguiente Tabla se presentan las soluciones dadas al problema anterior utilizando tres algoritmos diferentes: el algoritmo descrito en [2, 11, 12] (Carrera-Sagols), el algoritmo desarrollado en esta investigación utilizando las heurísticas descritas en el apéndice para aproximar el Problema Maestro

y los Subproblemas (Calderón-Vásquez Heurística) y el algoritmo desarrollado en esta investigación utilizando el programa `lp_solve` para determinar la solución del Problema Maestro y de los Subproblemas:

Algoritmo	Carrera-Sagols [12]	Calderón-Vásquez Heurística	Calderón-Vásquez `lp_solve`
Patrones	$\begin{pmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 2 \\ 3 \end{pmatrix}$	$\begin{pmatrix} 12 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 12 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 12 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 12 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 12 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 12 & 2 \end{pmatrix}$	$\begin{pmatrix} 12 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 12 \end{pmatrix}$
Copias por patrón	4742	(310, 121, 505, 1101, 666, 1085, 512)	(395, 206, 591, 1186, 752, 1170)
Copias sobrantes	5350	46	46
Espacio no utilizado	0	0	0

Tabla 5.3: Solución del problema anterior utilizando diferentes algoritmos.

Los datos presentados en la fila llamada "Copias sobrantes" representan el total de copias sobrantes de todos los gráficos después de realizar la impresión de todas las placas. Y los datos de la fila "Espacio no utilizado" representan todo el espacio no utilizado tomando en cuenta todas las planchas.

Para el caso del algoritmo Carrera-Sagols, se dispone únicamente de un molde. Así, el vector en la fila llamada "Patrones" según corresponda, rep-

resenta la distribución de los negativos de los gráficos en la plancha, por ejemplo, se debe colocar un negativo del primer gráfico y tres del cuarto. Gráficamente tenemos:

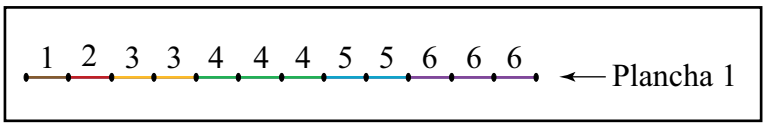


Figura 5.3: Solución gráfica dada por el algoritmo Carrera-Sagols, presentada en la Tabla 5.3.

En la figura anterior el segmento grande, formado por los pequeños segmentos de colores, representa la plancha. Cada uno de los colores representa un gráfico y el número que está arriba indica el gráfico que representa cada segmentito o bien cada color.

En los casos del algoritmo Calderón-Vásquez, la cantidad de moldes disponibles no es fija, en ambos casos se inició con 6 moldes, los cuales forman una matriz diagonal donde los elementos de la diagonal son todos igual a 12. Es decir, en cada molde hay 12 negativos de uno de los gráficos (un molde por gráfico) y a partir de esa distribución el algoritmo se analiza cual o cuales patrones es más conveniente agregar y cuales eliminar.

El algoritmo Calderón-Vásquez que utiliza la heurística, para resolver los problemas de programación lineal entera, da como resultado una solución con siete patrones los cuales están representados con las columnas de la matriz presentada en la columna Calderón-Vásquez (Heurística) en la Tabla 5.3. La representación gráfica de dicha solución se presenta en la siguiente figura:

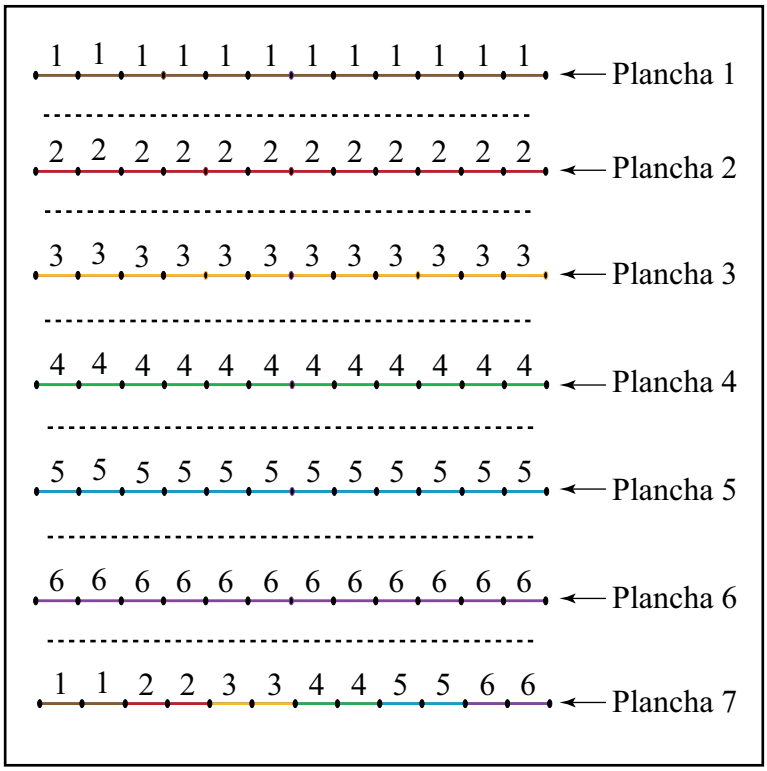


Figura 5.4: Solución gráfica dada por el algoritmo Calderón-Vásquez (Heurística), presentada en la Tabla 5.3.

La figura anterior presenta, de forma gráfica, las distribuciones correspondientes de los negativos en cada una de las planchas, según corresponda y según lo dado en la solución presentada en la Tabla 5.3 para el caso del algoritmo Calderón-Vásquez (Heurística). Los segmentos grandes que están separados por las lineas discontinuas representan cada una de las planchas. Así, por ejemplo, la quinta plancha tiene doce negativos del quinto gráfico y la séptima tiene dos negativos de cada uno de los gráficos.

El algoritmo Calderón-Vásquez que utiliza el método `lp_solve`, para resolver los problemas de programación lineal entera, da como resultado una solución con seis patrones. Siguiendo la solución presentada en la Tabla 5.3 en la columna Calderón-Vásquez (`lp_solve`) se tiene la siguiente representación gráfica:

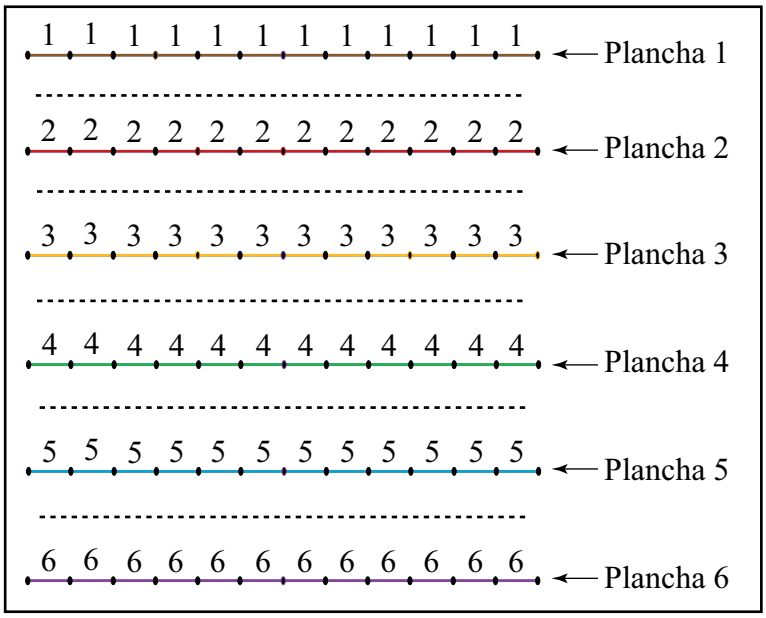


Figura 5.5: Solución gráfica dada por el algoritmo Calderón-Vásquez (`lp_solve`), presentada en la Tabla 5.3.

En la figura anterior se muestran seis moldes, separados por las lineas discontinuas, donde cada uno de ellos tiene doce negativos de un gráfico.

Por otro lado, note que en los tres casos se utiliza toda la capacidad del molde, pero la solución dada por el algoritmo Carrera-Sagols, en [12], da como resultado una mayor cantidad de copias sobrantes o copias que se sacan de

más, sin embargo, las soluciones dadas por los algoritmos Calderón-Vásquez necesita más cantidad de moldes disponibles. Lo anterior se da ya que dicho algoritmo tiene como objetivo conjunto minimizar la cantidad de copias que se sacan de más y el espacio no utilizado en cada plancha. Es importante observar que los dos aspectos antes mencionados tienen el mismo peso, en este caso es uno, se podría formular otro problema en el cuál cada uno de esos aspectos tenga un peso diferente, dependiendo del objetivo e intereses del problema, pero ese sería un problema aparte y completamente diferente al que se está presentado en este trabajo.

Considere la Tabla 5.4, donde se presentan los resultados del algoritmo Calderón-Vásquez (C-V) implementado de dos formas: utilizando la heurística mencionada anteriormente y utilizando el programa `lp_solve`. Se comparan los siguientes datos: cantidad de iteraciones que utilizó cada algoritmo para dar la solución, tiempo que tardó cada algoritmo en dar la solución, cantidad de patrones de la solución dada, cantidad de copias sobrantes de todos los gráficos después de realizar todas las impresiones y el espacio no utilizado en todas las planchas de las solución dada:

Cant. Graf.	C-V	Iter.	Tiempo (seg.)	Cant. Patr.	Cop. Sob.	Esp. no Util.
10	Heurística	11	0.382499731	10	14	121
	`lp_solve`	100000	1017.489059	7	60889	17
20	Heurística	9	0.301670673	19	6736	825
	`lp_solve`	100000	1112.938301	18	187972	69
30	Heurística	9	0.313607735	29	12791	1530
	`lp_solve`	100000	1615.869	27	119784	376
40	Heurística	8	0.27883935	38	23905	2157
	`lp_solve`	100000	3090.699979	33	196320	120
50	Heurística	18	0.697703898	47	30695	2180
	`lp_solve`	100000	2524.63115	47	291472	440
60	Heurística	5	0.365126084	57	30085	3709
	`lp_solve`	100000	4315.212543	57	377545	313

Tabla 5.4: Resultados de diferentes problemas utilizando el algoritmo C-V en una dimensión.

En la Tabla 5.4 se presentan las soluciones dadas a problemas con diferente cantidad de gráficos, en cada uno de los casos se resuelve el mismo problema con ambos algoritmos. Note que en todos los casos se tiene que la cantidad de patrones de la solución dada utilizando `lp_solve` es menor o igual que la cantidad de patrones cuando se utiliza la heurística, consecuentemente el espacio no utilizado en la solución de `lp_solve` también es menor o igual que el de la solución de la heurística, sin embargo, la cantidad

de copias que se sacan de más utilizando `lp_solve` es mucho mayor que las copias sobrantes utilizando la heurística.

Si se observa cuidadosamente la cantidad de patrones dados en la solución de los problemas con 50 y 60 gráficos, se puede notar que ambos algoritmos dan como resultado la misma cantidad de patrones. Sin embargo, la cantidad de copias que se sacan de más para el caso de la heurística siempre es menor que cuando se utiliza `lp_solve` y el espacio no utilizado en la solución dada por la heurística es mayor que en la solución dada por el algoritmo que utiliza `lp_solve`. Ya que a pesar que la cantidad de patrones es la misma no necesariamente los patrones van a ser los mismos, en cada uno de los casos, y eso también está generando una gran diferencia en los resultados obtenidos por los diferentes métodos.

Otro resultado es el relacionado con los tiempos de ejecución, para obtener los resultados anteriores, el tiempo de ejecución utilizando la heurística siempre es mucho menor que utilizando `lp_solve`. Además, en todos los casos, el algoritmo que utiliza `lp_solve` siempre llega a la cantidad máxima de iteraciones permitidas y, posiblemente, sin haber llegado al óptimo. Las cifras de los tiempos de ejecución nos indican que conforme vaya aumentando la cantidad de gráficos del problema el algoritmo que utiliza `lp_solve` va a tomar tiempos demasiado grandes, hasta llegar al punto que el tiempo necesario para que dicho algoritmo resuelva un problema con muchos gráficos no es racional o no puede ser considerado, comparado con la realidad del problema.

Es importante mencionar que cuando $n > 80$, donde n es la cantidad de gráficos en el problema, no es posible determinar la solución a dicho proble-

ma utilizando el método Calderón-Vásquez (`lp_solve`), por ejemplo cuando $n = 85$ tal algoritmo estuvo ejecutándose alrededor de 24 horas y no se logró encontrar solución óptima.

5.2.2. Dos dimensiones

La siguiente Tabla presenta los resultados del algoritmo Calderón-Vásquez (C-V) implementado de dos formas: utilizando la heurística (Ver apéndice) y utilizando el programa `lp_solve`. Se comparan los siguientes datos: cantidad de iteraciones que utilizó cada algoritmo para dar la solución, tiempo que tardó cada algoritmo en dar la solución, cantidad de patrones de la solución dada, cantidad de copias sobrantes de todos los gráficos después de realizar todas las impresiones y el espacio no utilizado, en unidades cuadradas, en todas las planchas de las solución dada:

Cant. Graf.	C-V	Iter.	Tiempo (seg.)	Cant. Patr.	Cop. Sob.	Esp. no Util.
10	Heurística	11	0.649715746	10	24	65584
	`lp_solve`	100000	2196.876149	9	42818	55904
20	Heurística	16	1.208376693	20	38	135563
	`lp_solve`	100000	2123.7285	16	225946	108657
30	Heurística	7	0.465283507	30	9	264244
	`lp_solve`	100000	1575.512147	24	76130	157078
40	Heurística	12	0.912720891	40	13	337606
	`lp_solve`	100000	3362.698513	36	415686	243283
50	Heurística	24	2.09572732	50	92	375583
	`lp_solve`	100000	6424.800718	43	138093	301820

Tabla 5.5: Resultados de diferentes problemas utilizando el algoritmo C-V en dos dimensiones.

Similarmente al caso de una dimensión, en la Tabla 5.5 se presentan las soluciones dadas a problemas con diferente cantidad de gráficos, en cada uno de los casos se resuelve el mismo problema con ambos algoritmos. Y se tienen los siguientes resultados:

- La cantidad de patrones de la solución dada utilizando `lp_solve` es menor o igual que la cantidad de patrones cuando se utiliza la heurística. Así el espacio no utilizado, en unidades cuadradas, en la solución de `lp_solve` también es menor o igual que el de la solución de la heurística.

- La cantidad de copias que se sacan en exceso utilizando `lp_solve` es mucho mayor que las copias sobrantes utilizando la heurística.
- Para problemas con muchos gráficos el tiempo de ejecución que toma el algoritmo con `lp_solve` podría no ser accesible o compatible con la realidad del problema.

Notablemente, el algoritmo que utiliza la heurística toma menos tiempo en determinar una solución entera al problema comparado con el algoritmo que utiliza `lp_solve`. Debido a que en las técnicas utilizadas por `lp_solve`, específicamente la de Ramificar y Acotar, consumen mucho tiempo y mucha memoria en la búsqueda de la solución, el método resuelve el mismo problema en diferentes regiones, cada vez más pequeñas, con el fin de encontrar la solución entera, es decir, resuelve el mismo problema varias veces.

5.3. Variación de resultados

5.3.1. Una dimensión

Considere los siguientes tres problemas en una dimensión, para los cuales se estableció el tamaño de la plancha $L = 100$ y la cantidad de gráficos $n = 40$, donde los datos l y d están dados en las Tablas 5.6 y 5.7, según corresponda a cada problema:

$$l = (25, 32, 22, 31, 37, 23, 28, 23, 37, 32, 22, 28, 26, 21, 39, 26, 38, 35, 21, 34...$$
$$..,29, 39, 28, 32, 37, 34, 38, 32, 39, 33, 34, 21, 39, 31, 40, 28, 36, 31, 35, 35)$$

$$d = (31116, 26286, 27427, 41340, 30453, 25012, 42742, 31301, 35700, 38012...$$
$$..,31048, 48174, 43895, 47357, 25304, 31140, 45639, 48977, 47372, 43459...$$
$$..,48735, 34979, 39248, 46062, 37509, 28741, 44052, 34707, 32860, 49088...$$
$$..,29662, 43865, 47956, 49903, 34313, 48934, 43917, 49753, 49686, 44231)$$

Tabla 5.6: Problema #1 en una dimensión con 40 gráficos.

Al resolver el problema de la Tabla 5.6 el algoritmo `UnaDimen` (página 31) tardó 0.365696339 segundos, aproximadamente, en resolver el problema con un total de 8 iteraciones y dando como resultado una solución con 37 patrones diferentes.

$$l = (38, 34, 28, 33, 35, 22, 25, 22, 38, 35, 31, 31, 21, 35, 24, 28, 30, 22, 26, 37, ...$$
$$..,31, 36, 38, 24, 30, 28, 32, 21, 36, 25, 35, 27, 21, 30, 29, 35, 23, 36, 34, 36)$$

$$d = (29672, 29260, 33385, 32402, 37597, 32523, 44420, 42869, 43080, 35587...$$
$$..,43641, 46901, 48605, 41771, 49029, 32240, 44821, 41786, 27295, 31145...$$
$$..,47159, 26372, 36250, 30371, 29419, 40324, 30070, 39482, 39960, 30442...$$
$$..,47899, 32673, 48643, 44312, 37425, 28393, 40747, 44329, 43994, 40800)$$

Tabla 5.7: Problema #2 en una dimensión con 40 gráficos.

Por otro lado, para el problema de la Tabla 5.7 el mismo algoritmo `UnaDimen` (página 31) tardó 1.031924588 segundos, aproximadamente, en resolver el problema con un total de 23 iteraciones y, en este caso, se obtuvo una solución con 35 patrones diferentes.

$$l = (32, 29, 34, 24, 25, 33, 36, 38, 34, 35, 36, 24, 21, 25, 33, 22, 27, 21, 23, 39, ...$$
$$..,25, 28, 29, 27, 21, 23, 24, 23, 28, 34, 34, 30, 25, 21, 35, 32, 30, 37, 28, 38)$$

$$d = (35321, 25850, 39254, 27969, 29945, 38335, 42436, 45842, 49349, 25257, ...$$
$$..,45053, 46969, 25393, 29281, 38145, 26190, 33080, 25849, 47734, 46928, ...$$
$$..,29525, 48078, 47096, 32796, 38121, 41985, 26632, 27695, 34004, 34002, ...$$
$$..,35298, 28947, 43345, 32257, 27616, 38619, 44547, 42819, 41669, 30432)$$

Tabla 5.8: Problema #3 en una dimensión con 40 gráficos.

Para el problema de la Tabla 5.8 el algoritmo `UnaDimen` (página 31) tardó 3781.207789 segundos, aproximadamente, en dar una solución aproximada al problema con un total de 100000 iteraciones, es decir, alcanzó el número máximo de iteraciones permitidas por el algoritmo y se obtuvo una solución con 34 patrones diferentes.

En las últimas tres Tablas 5.6, 5.7 y 5.8 se presentan problemas similares, en el sentido que tienen la misma cantidad de gráficos y que las dimensiones y las demandas de cada uno de ellos están dentro de los mismos rangos. Sin embargo, los tiempos de ejecución son distintos.

En la Tabla 5.9 se muestra un resumen de los datos de los gráficos y de la solución presentados en cada uno de los problemas anteriores.

Problema	#1 - Tabla 5.6	#2 - Tabla 5.7	#3 - Tabla 5.8
Cantidad de iteraciones	8	23	100000
Tiempo de ejecución	0.365696339	1.031924588	3781.207789
Coeficiente de variación de las dimensiones	18.91 %	18.47 %	19.16 %
Coeficiente de variación de las demandas	20.65 %	18.22 %	21.57 %

Tabla 5.9: Coeficiente de variación de las dimensiones y demandas en los problemas de las Tablas 5.6, 5.7 y 5.8.

Así, se podría decir que la gran diferencia entre los tiempos de ejecución en la resolución de cada uno de los problemas presentados en las Tablas 5.6, 5.7 y 5.8 se debe a la variación de los datos.

Note que el tiempo de ejecución del Problema #1 y del Problema #2 son muy similares y lo mismo ocurre con sus tiempos de ejecución, por lo que entre esos dos poblemas no se podría establecer una relación muy certera respecto a los tiempos de ejecución y a los coeficientes de variación ya que, por sus coeficientes de variación, los datos de los problemas son muy similares. Las diferencias significativas se presentan entre el Problema #1 y el Problema #3, lo mismo que entre el Problema #2 y el Problema #3, donde la diferencia entre los tiempos de ejecución es mucho mayor. De esa manera, con los resultados de la Tabla 5.9 se puede intuir que para este caso, en una dimensión, si el coeficiente de variación de las dimensiones de los gráficos es

menor entonces el tiempo que tarda el algoritmo en encontrar una solución también va a ser menor.

5.3.2. Dos dimensiones

Observe los siguientes tres problemas, ahora en dos dimensiones, para las planchas se estableció largo $A = 100$ y ancho $B = 100$, con $n = 40$ (a contiene los anchos, b los largos y d las demandas de los gráficos) y note que se tiene un comportamiento similar al caso en una dimensión:

$$a = (30, 39, 39, 27, 27, 36, 38, 29, 25, 24, 27, 30, 32, 22, 28, 29, 35, 36, 30, 29...$$
$$..,34, 29, 24, 36, 33, 21, 20, 36, 24, 37, 39, 33, 39, 22, 27, 24, 28, 40, 25, 33)$$

$$b = (29, 34, 24, 35, 23, 38, 30, 21, 26, 27, 27, 20, 27, 32, 32, 27, 38, 33, 23, 20...$$
$$..,38, 40, 31, 33, 36, 24, 36, 38, 22, 30, 24, 39, 40, 26, 21, 35, 38, 34, 34, 32)$$

$$d = (44142, 45686, 27915, 29944, 27328, 44986, 28097, 25769, 44983, 38130...$$
$$..,44125, 32877, 33662, 33214, 47546, 36118, 41594, 35834, 26996, 44810...$$
$$..,41329, 31558, 42967, 34741, 41719, 30482, 28737, 48712, 28542, 35690...$$
$$..,44221, 36043, 49843, 33739, 27678, 25773, 44387, 40735, 41332, 30674)$$

Tabla 5.10: Problema #1 en dos dimensiones con 40 gráficos.

Al resolver el problema de la Tabla 5.10 el algoritmo `DosDimen` (página 36) tardó aproximadamente 0.936680849 segundos en resolver el problema

en 12 iteraciones y, en este caso, se obtuvo una solución con 39 patrones diferentes.

$$a = (40, 39, 32, 40, 24, 40, 40, 24, 31, 32, 32, 39, 26, 34, 23, 37, 32, 37, 30, 33...$$
$$..,38, 21, 22, 21, 35, 29, 32, 32, 39, 27, 39, 29, 32, 25, 20, 28, 27, 33, 36, 36)$$

$$b = (30, 20, 29, 29, 32, 21, 26, 23, 21, 25, 22, 38, 33, 25, 30, 22, 34, 20, 27, 22...$$
$$..,27, 34, 33, 37, 36, 27, 36, 23, 21, 30, 36, 38, 40, 40, 29, 20, 26, 39, 23, 32)$$

$$d = (33477, 29297, 36785, 49821, 30563, 34346, 43570, 49362, 38872, 41516...$$
$$..,36449, 43345, 35157, 36214, 39093, 48547, 40419, 26358, 27923, 36206...$$
$$..,31178, 38395, 25853, 46881, 33343, 38270, 42283, 48745, 40267, 36675...$$
$$..,42222, 31816, 43450, 48151, 26705, 44995, 45465, 33422, 32591, 44777)$$

Tabla 5.11: Problema #2 en dos dimensiones con 40 gráficos.

Por otro lado, para el problema de la Tabla 5.11 el mismo algoritmo `DosDimen` (página 36) tardó aproximadamente 2.650484279 segundos en resolver el problema en 32 iteraciones y dando como resultado una solución con 40 patrones diferentes.

$$a = (24, 33, 40, 38, 21, 30, 23, 22, 34, 35, 32, 31, 25, 21, 23, 32, 39, 29, 23, 35, ...$$
$$..,20, 26, 37, 34, 39, 34, 21, 26, 30, 34, 39, 25, 25, 24, 38, 24, 35, 37, 28, 30)$$

$$b = (40, 34, 33, 30, 20, 34, 28, 33, 24, 37, 27, 23, 30, 38, 20, 37, 23, 31, 23, 33, ...$$
$$..,20, 21, 32, 40, 22, 29, 35, 35, 27, 39, 39, 24, 28, 38, 20, 40, 29, 32, 23)$$

$$d = (37832, 49775, 48122, 26754, 48194, 44923, 29833, 45100, 29010, 26549, ...$$
$$..,45061, 27028, 35104, 27134, 35621, 36748, 40047, 47706, 27141, 27006, ...$$
$$..,45752, 33089, 32138, 36354, 39023, 29702, 42069, 44017, 42861, 48181, ...$$
$$..,26840, 48447, 38106, 42225, 30767, 31967, 43860, 39243, 45867, 40699)$$

Tabla 5.12: Problema #3 en dos dimensiones con 40 gráficos.

Para el problema de la Tabla 5.12 el algoritmo `DosDimen` (página 36) tardó 5995.681247 segundos, aproximadamente, en dar una solución aproximada al problema para un total de 100000 iteraciones, es decir, alcanzó el límite de iteraciones establecido en el algoritmo y se obtuvo una solución con 40 patrones diferentes.

En las Tablas 5.10, 5.11 y 5.12 se presentan problemas similares, con la misma cantidad de gráficos, con las dimensiones y las demandas de cada uno de ellos están dentro de los mismos rangos. Pero, los tiempos de ejecución, en este caso, también son diferentes.

La Tabla 5.13 muestra un análisis de los datos en cada uno de los problemas presentados anteriormente:

Problema	#1 - Tabla 5.10	#2 - Tabla 5.11	#3 - Tabla 5.12
Cantidad de iteraciones	12	32	100000
Tiempo de ejecución	0.936680849	2.650484279	5995.681247
Coeficiente de variación de los anchos	18.97 %	19.26 %	20.74 %
Coeficiente de variación de los largos	20.35 %	21.95 %	21.63 %
Coeficiente de variación de las demandas	19.81 %	17.87 %	20.29 %

Tabla 5.13: Coeficiente de variación de los anchos, los largos y las demandas en los problemas de las Tablas 5.10, 5.11 y 5.12.

Similar al caso en una dimensión, se podría decir que la diferencia entre los tiempos de ejecución de los problemas presentados en las Tablas 5.10, 5.11 y 5.12 se debe a la variación de los datos. Así, conforme el coeficiente de variación de las dimensiones de los gráficos aumente también aumentará el tiempo que tarda el algoritmo en encontrar una solución.

Se puede inferir que el tiempo de ejecución de un problema depende de la variación en los datos del mismo. Un problema con mayor variación en los datos indica que los mismos presentan mayores diferencias, esto hace que el algoritmo tome más tiempo en determinar los diferentes patrones. Por el contrario, si la variación de los datos es menor los datos del problema son similares y así, para el algoritmo es más "sencillo" encontrar los patrones.

5.4. Tiempos de ejecución

Las siguientes gráficas representan los tiempos de ejecución que tomó el algoritmo Calderón-Vásquez (Heurística) en resolver problemas con diferentes cantidades de gráficos.

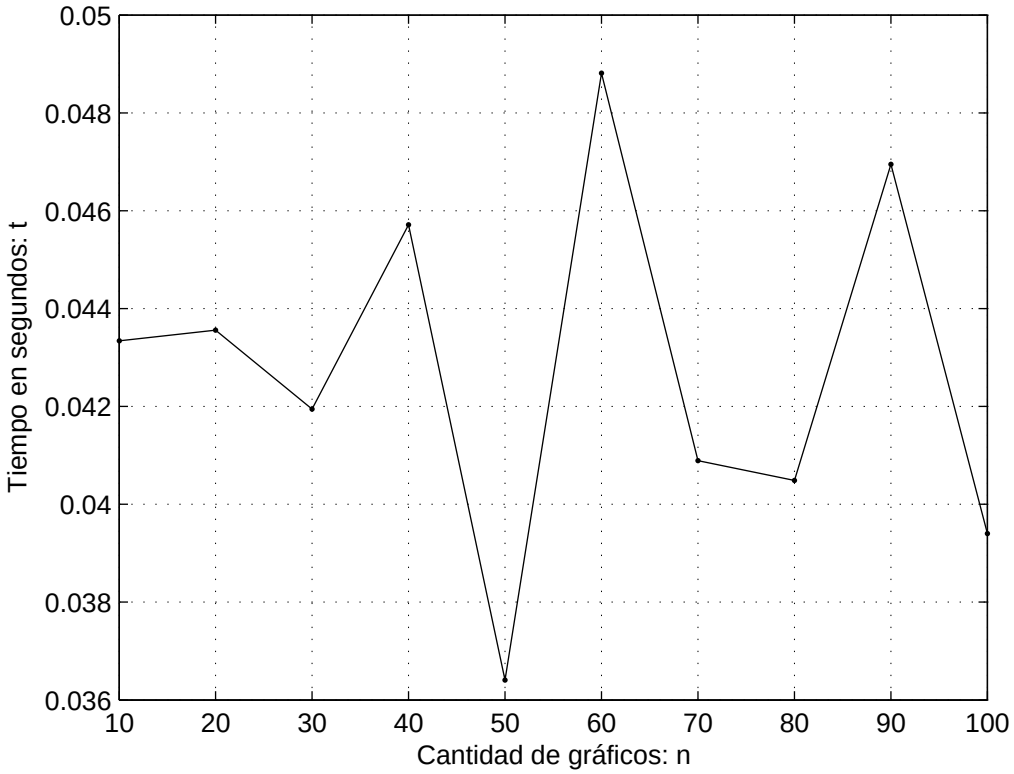


Figura 5.6: Una dimensión: Tiempo de ejecución de una iteración para problemas con 10, 20, ..., 100 gráficos y moldes de capacidad 100.

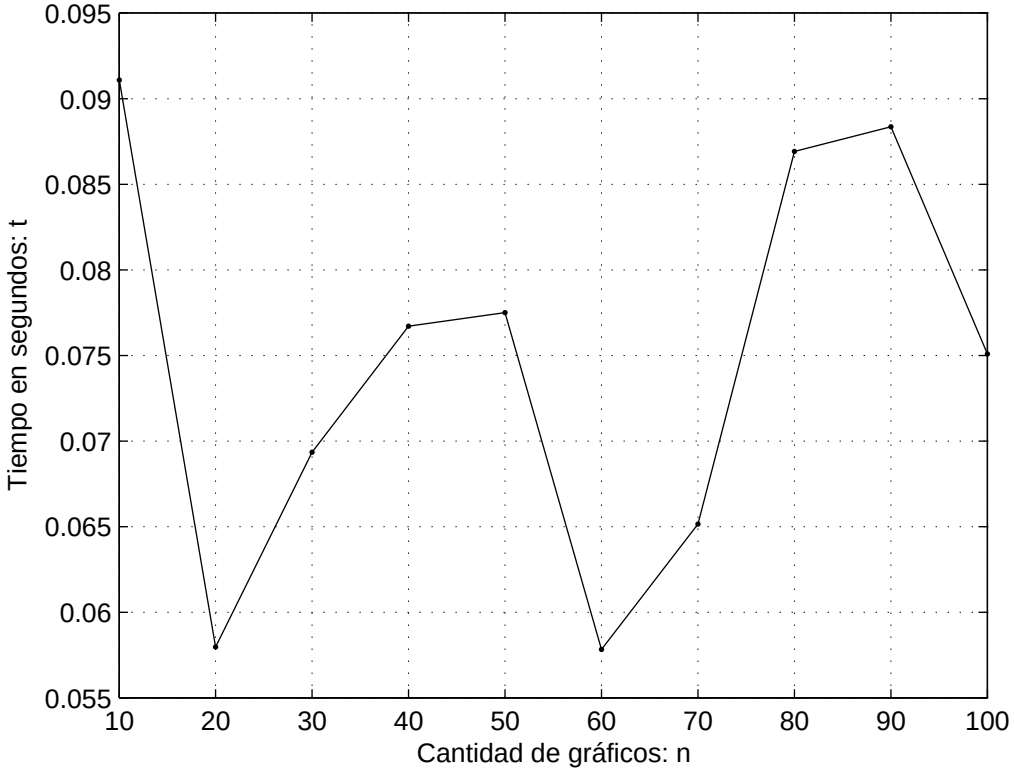


Figura 5.7: Dos dimensiones: Tiempo de ejecución de una iteración para problemas con 10, 20, ..., 100 gráficos y moldes de capacidad 100×100.

En las Figuras 5.6 y 5.7 se puede notar que la variación en los tiempos de ejecución de una iteración en cada uno de los casos es muy grande. Hasta el momento no se puede establecer un patrón o generalizar de forma gráfica el comportamiento de los tiempos. Sin embargo, se puede intuir que la variación entre las dimensiones y demandas de cada uno de los problemas tiene mayor influencia que, por ejemplo, cuando se genere un problema de 500 gráficos bajo las mismas condiciones: dimensiones en el rango $[20, 40]$ y demandas en

$[25000, 50000]$.

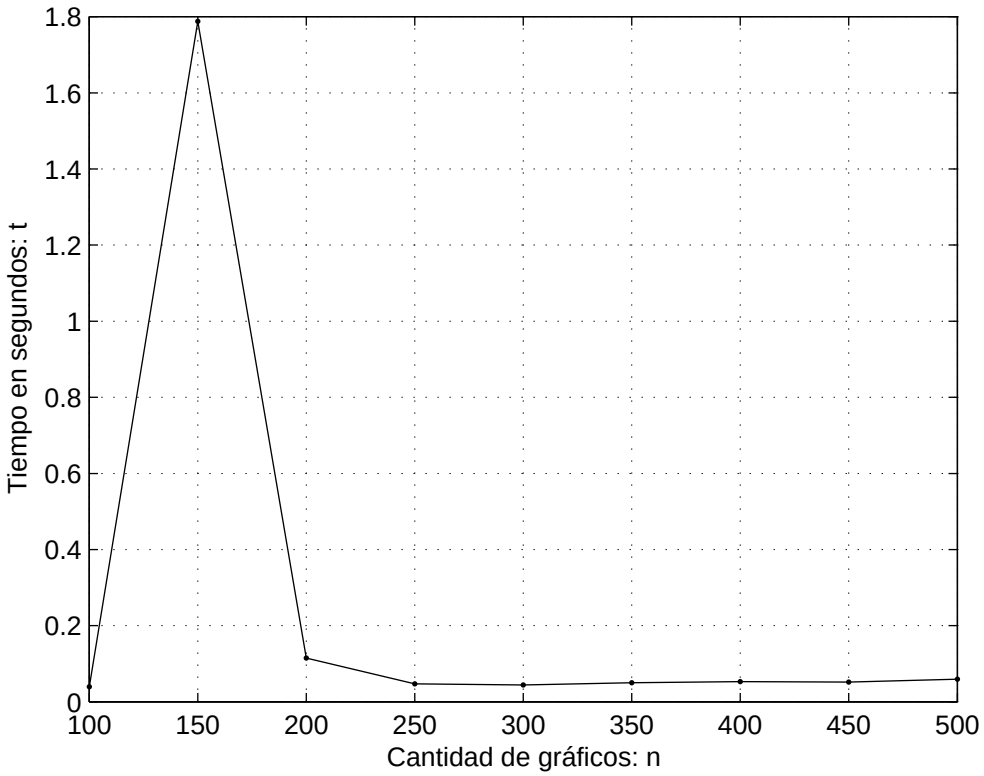


Figura 5.8: Una dimensión: Tiempo de ejecución de una iteración para problemas con 100, 150, ..., 500 gráficos y moldes de capacidad 100.

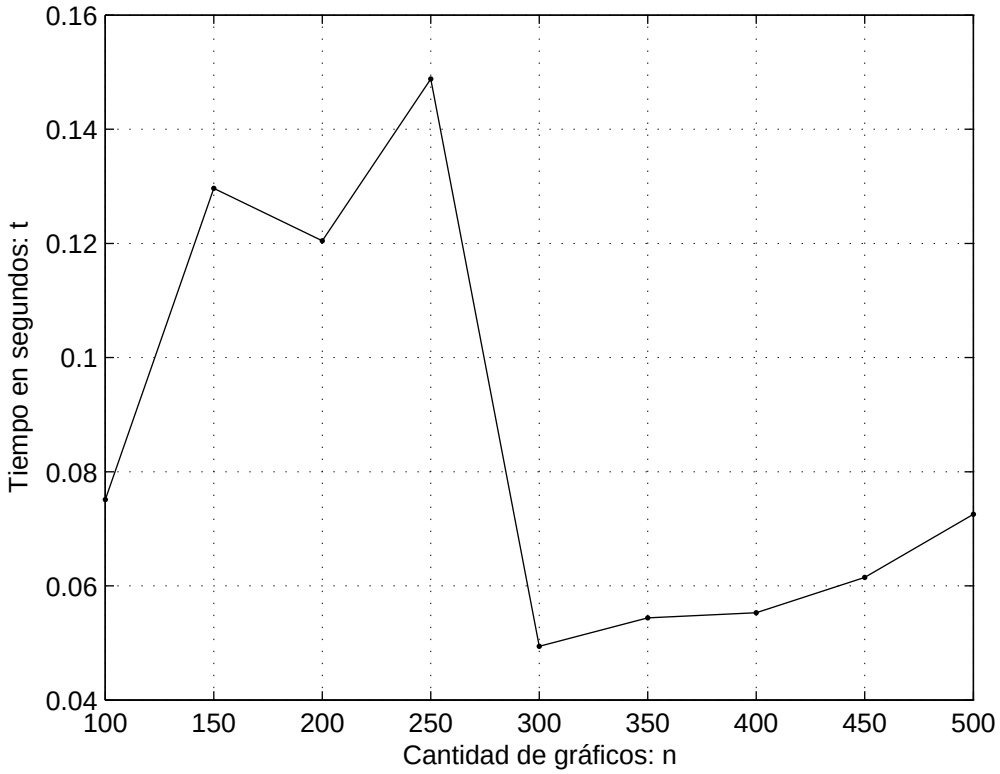


Figura 5.9: Dos dimensiones: Tiempo de ejecución de una iteración para problemas con 100, 150, ..., 500 gráficos y moldes de capacidad 100×100.

Note en las Figuras 5.8 y 5.9 que conforme la cantidad de gráficos va aumentando la variación antes mencionaba va disminuyendo y con eso el tiempo de ejecución de una iteración se va estabilizando. De esa forma se podría intuir que va tomando un comportamiento lineal o exponencial.

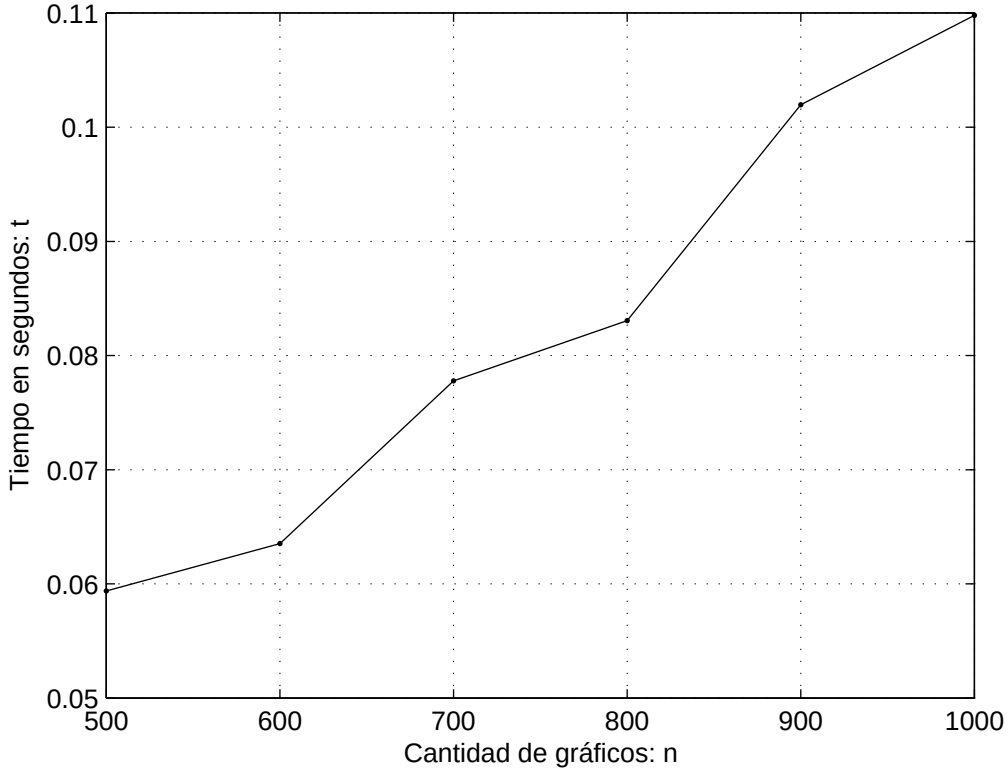


Figura 5.10: Una dimensión: Tiempo de ejecución de una iteración para problemas con 500, 600, ..., 1000 gráficos y moldes de capacidad 100.

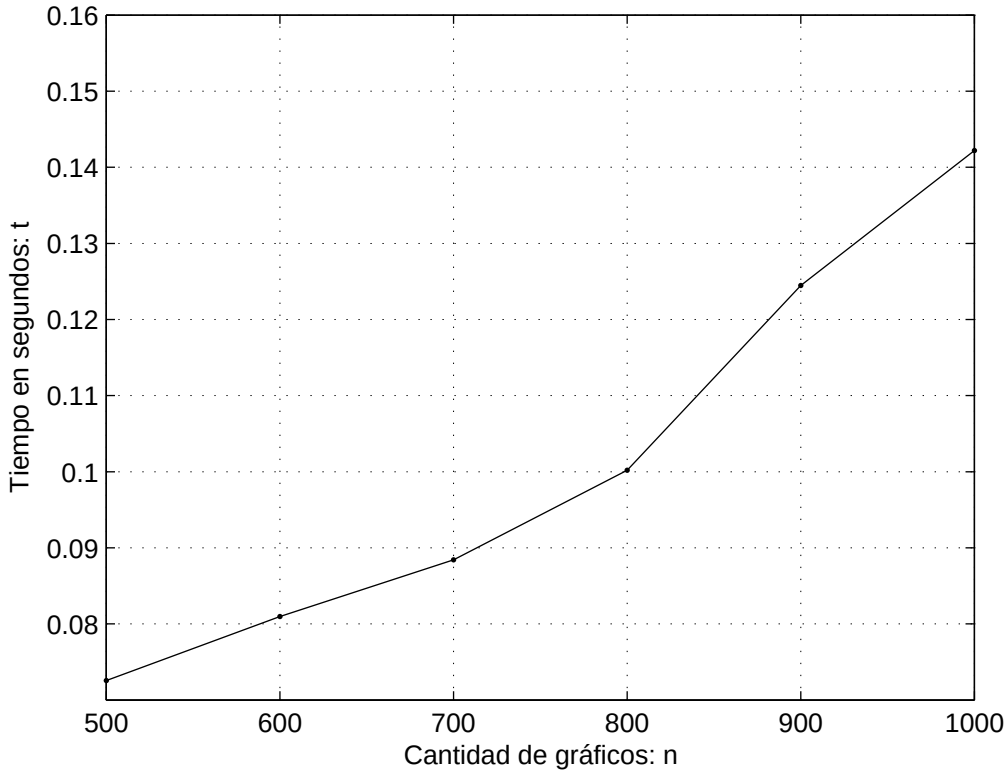


Figura 5.11: Dos dimensiones: Tiempo de ejecución de una iteración para problemas con 500, 600, ..., 1000 gráficos y moldes de capacidad 100×100.

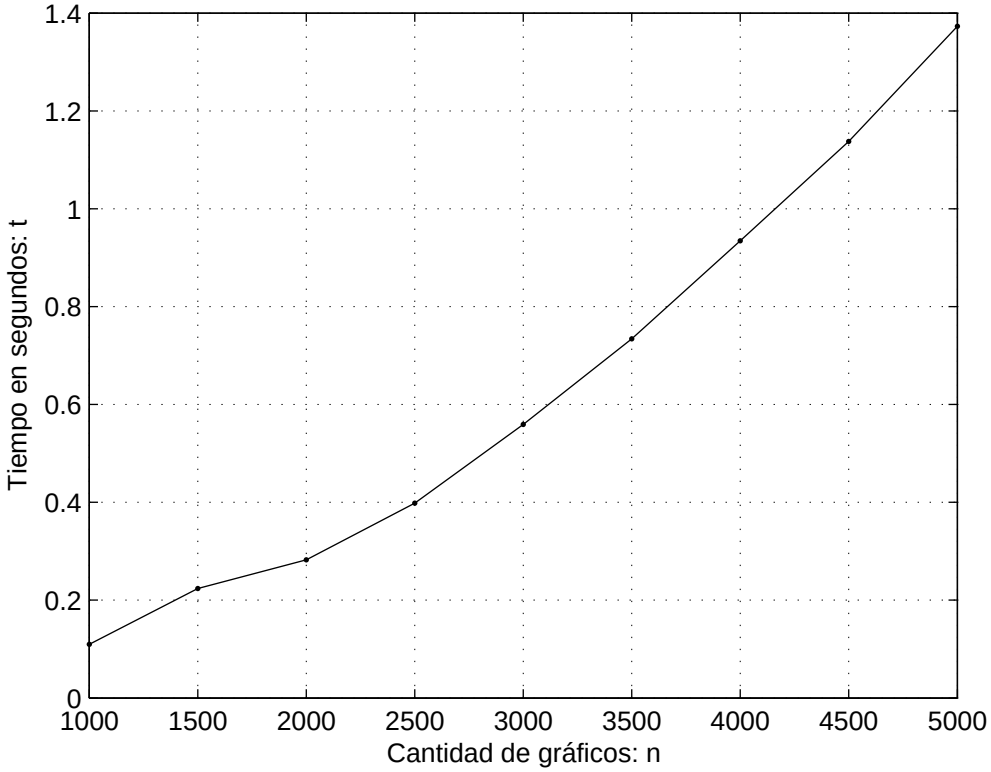


Figura 5.12: Una dimensión: Tiempo de ejecución de una iteración para problemas con 1000, 1500, ..., 5000 gráficos y moldes de capacidad 100.

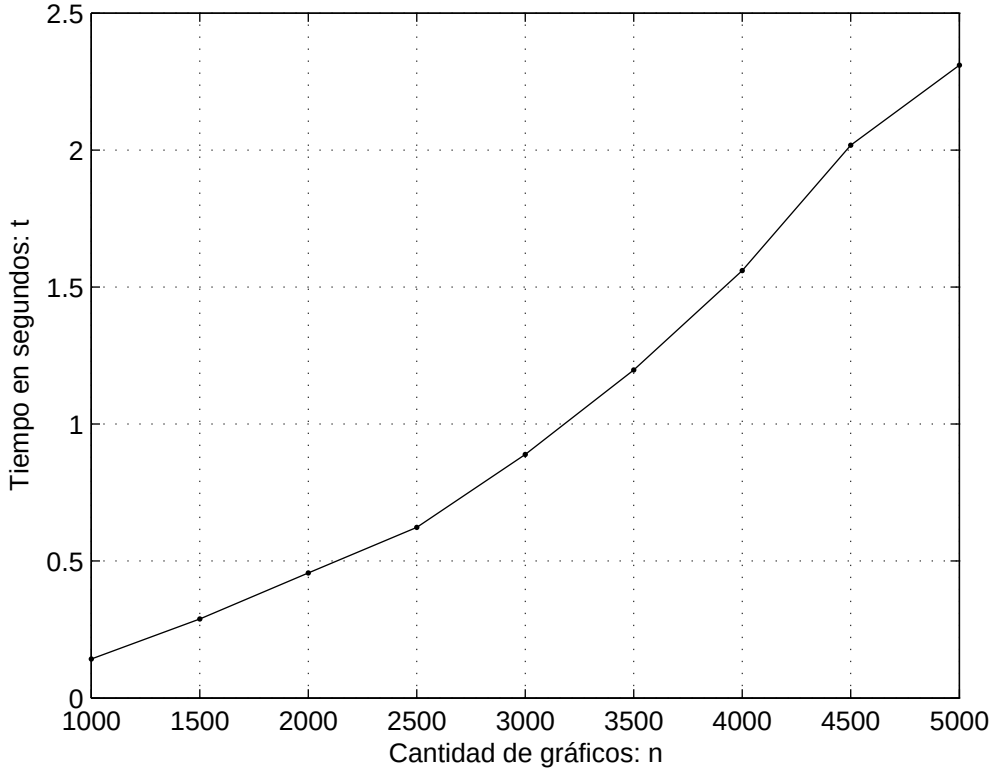


Figura 5.13: Dos dimensiones: Tiempo de ejecución de una iteración para problemas con 1000, 1500, ..., 5000 gráficos y moldes de capacidad 100×100.

Ya en las Figuras 5.10, 5.11, 5.12 y 5.13 observe que el comportamiento del tiempo está estabilizado en el sentido que se puede notar el aumento exponencial en los tiempos dependiendo de la cantidad de gráficos que son considerados.

En general, se tiene la siguiente figura:

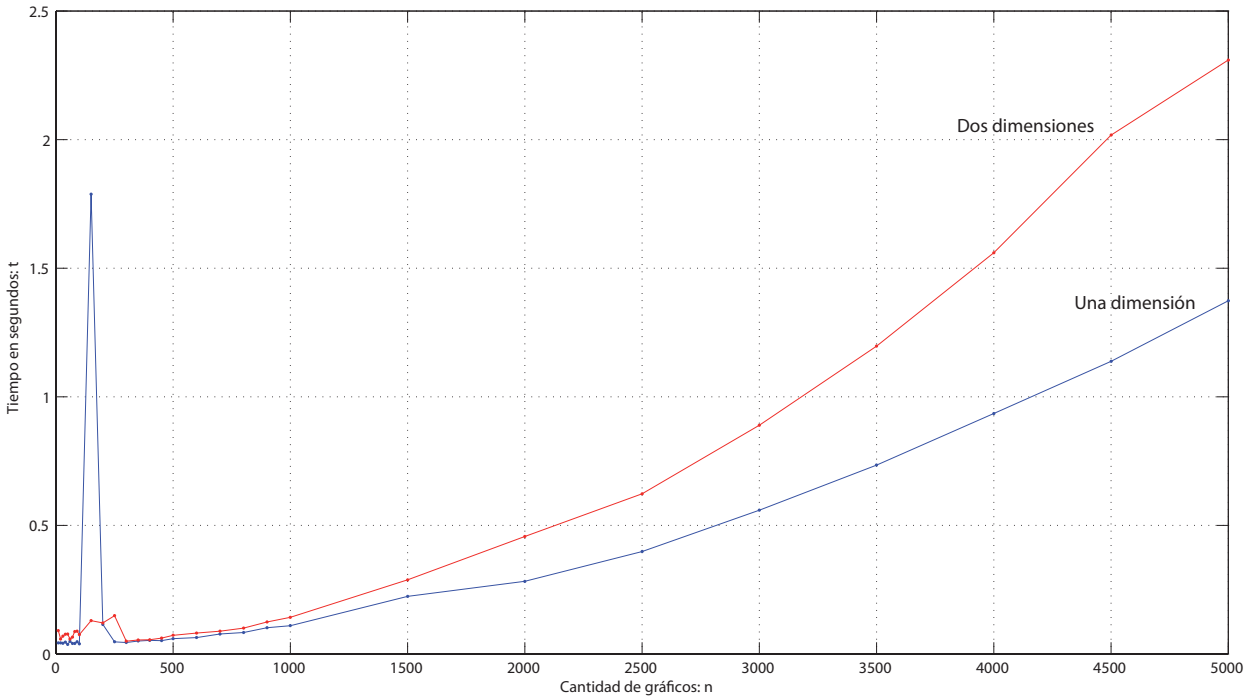


Figura 5.14: Tiempo de ejecución de una iteración.

En el gráfico de la Figura 5.14 se puede observar el tiempo de ejecución tiene un comportamiento "casi" exponencial en ambos algoritmos, `UnaDim` y `DosDim`. Además, de la inestabilidad en los tiempos de ejecución para los problemas con menor cantidad de gráficos, debido a las diferentes variaciones en los datos de cada uno de los problemas.

Notablemente el tiempo de ejecución del algoritmo en una dimensión es

menor al del algoritmo en dos dimensiones, debido a que en el algoritmo en una dimensión siempre resuelve un subproblema por iteración, mientras que el algoritmo en dos dimensiones en el mejor de los casos resuelve dos subproblemas y en el peor de los casos resuelve $n + 1$ subproblemas, donde n es la cantidad de gráficos del problema.

Capítulo 6

Conclusiones y trabajos futuros

6.1. Conclusiones

Como resultado de esta investigación se presentan las siguientes conclusiones:

- Se logró resolver el problema en una dimensión. Considerando gráficos de diferentes tamaños y tomando en cuenta más de un molde, sin necesidad que se deba colocar un negativo de cada gráfico en cada molde.
- Se logró implementar un algoritmo que brinde una solución en dos dimensiones, con características similares al de una dimensión.
- Para que la solución dada por el algoritmo sea factible se deben tener al menos n moldes disponibles. Además, el algoritmo no trabaja con una cantidad prefijada de moldes.

- Gráficamente se puede notar que el tiempo de ejecución, por iteración, crece exponencialmente con respecto a la cantidad de gráficos del problema.

- El tiempo de ejecución del algoritmo completo depende directamente de los datos del problema y no de la cantidad de gráficos en el problema.

- El tiempo de ejecución del algoritmo en una dimensión es menor al algoritmo en dos dimensiones, debido a la cantidad de subproblemas que se resuelven en cada uno de los casos.

- Para el caso de dos dimensiones podría no aprovecharse al máximo el espacio del molde, debido a las líneas imaginarias con las que se trabaja para resolver el problema.

- La utilización de la heurística para aproximar la solución entera de cada uno de los problemas de programación lineal permite acelerar el proceso y el tiempo de ejecución del algoritmo con dicha heurística es mucho menor comparado con el tiempo que tarda el algoritmo utilizando técnicas como Ramificar y Acotar para determinar la solución de un problema de programación entera.

- La solución dada por el algoritmo que utiliza el método de Ramificar y Acotar para determinar la solución de un problema de programación entera da como resultado un gran número de copias sobrantes comparado con el algoritmo que utiliza heurísticas para aproximar la solución de un problema de programación entera. Sin embargo, la cantidad de patrones en la solución dada por el algoritmo que utiliza Ramificar y

Acotar es siempre menor o igual que el algoritmo que utiliza la heurística y consecuentemente, el espacio no utilizado en las planchas también va a ser menor o igual. Además, este algoritmo sólo se puede aplicar para un número de gráficos menor o igual a 80.

6.2. Trabajos futuros

Como trabajo adicional relacionado al problema de la industria editorial se sugiere:

- Los resultados presentados pueden ser extendidos a tres dimensiones, analizando la posibilidad de resolver el problema por partes de una manera similar a como se hizo en dos dimensiones.
- Además, se podrían implementar técnicas como Programación Dinámica en la búsqueda de una solución al problema de la industria editorial donde la cantidad de planchas disponibles sea fija, pero no necesariamente una.
- Un trabajo interesante también sería involucrar los costos de crear las planchas y producir los negativos de los gráficos. Además, establecer los presupuestos máximos disponibles en el problema.

Bibliografía

[1] Ahuja, Ravindra K., Magnanti, Thomas L., Orlin, James B. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall. New Jersey, Estados Unidos, 1993.

[2] Carrera Retana, Luis Ernesto. *Algoritmo para encontrar el óptimo a una simplificación de un problema combinatorio presente en la industria editorial.* Tesis de Maestría. Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional. México, D.F., 2007.

[3] Castillo, Enrique, Conejo, Antonio J., Pedregal, Pablo, García, Ricardo y Alguaci, Natalia. *Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia*. Ciudad Real, España, 2002.

[4] Desaulniers, Guy, Desrosiers, Jacques and Solomon, Marius M. *Column Generation*. Springer Science+Bussines Media, Inc, 2005

[5] Ehrgott, Matthis and Tind, Jorgen. *Column Generation in Integer Programming with Applications in Multicriteria Optimization*. March 27, 2007.

[6] Karelahti, Janne. *Solving the cutting stock problem in the steel industry.* Master's thesis submitted in partial fulfillment of the requiriments dor the degree of Master of Science in Technology. Espoo, Finlandia, 2002.

[7] Martello, Silvano and Toth, Paolo. *Knapsack Problems: Algorithms and Computer Implementations.* Wiley, NY, 1990.

[8] Maya, Pablo Andrés. *Revista de la Facultad de Ingeniería de la Universidad Antioquía N.°46.* Column Generations Algorithm: A revision from its application to Student Assignation Problem (2008) 145-157.

[9] Nocedal Jorge and Wright, Stephen J. *Numerical Optimization.* Springer series in operations research. New York, 1999.

[10] Papadimitriou, Christos H. and Steiglitz Kenneth. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications, INC. Minesota, NY, 1998.

[11] Sagols, Feliú and Carrera-Retana, Luis. *A new knapsack problem variant.* Working paper, 2008.

[12] Sagols, Feliú, Carrera, Luis y Flores, Alejandro. *Diseño eficiente de moldes: un heurístico.* Artículo de Trabajo-No publicado, 2009.

[13] Smith, Jeffrey D. *Desing and Analysis of Algorithms.* PWS-KENT Publishing Company. Boston, Massachusetts, 1951.

[14] Stidsen, Thomas. *Column Generation: Cutting Stock - A very applied method.* Technical University of Denmark, Working Presentation (Informatics and Mathematical Modeling).

[15] Teo Chung-Piaw (NUS). *Column Generation*. Working Paper. Singapure, 2003.

[16] Wayne L. Winston. *Operations Research: Applications and Algorithms*. Duxbury Press. Third Edition. Belmont, California, 1993.

[17] Yaodong Cui. *Simple blocks patterns for two-dimensional cutting problem*. Mathematical and Computing Modeling 45(2007) 943-953.

[18] Yaodong Cui and Yuli Yang. *A heuristic for the one-dimensional cutting stock problem with usable leftover*. European Journal of Operational Research 204(2010) 245-250.

[19] Yusuke Shiomi, Massao Sugi, Tsuyoshi Okubo, Masashi Yamamoto, Hiroshi Kojima and Kazuyoshi Inoue *The solution of 2-Dimensional Rectangular Cutting Stock Problem Considering Cutting Process*. Proceedings of the 3rd Annual IEEE Conference on Automation Science and Engineering Scottsdale, AZ, USA, Sept 22-25, 2007.

Apéndice A

Algoritmos

A.1. Problema de programación lineal

Este algoritmo aproxima la solución de un problema de programación lineal de la forma

$$\begin{aligned} \text{minimizar} \quad & f^T x \\ \text{Sujeto a:} \quad & Ax \geq b \\ & x \geq 0 \end{aligned} \tag{A.1}$$

utilizando la función de MatLab `linprog`, donde f es el vector que contiene los coeficientes de las variables de decisión.

Sin embargo, es posible que la solución dada por esa función no sea entera y que además dicha solución tenga varias cifras decimales. Así, el algoritmo siempre va a redondear la solución dada a dos cifras decimales.

Algoritmo A.1: Aproximación a un problema de programación lineal

`[x] = Aprox(f,A,b)`

Entrada: $f \in \mathbb{R}^m$; $A \in \mathbb{R}^{n \times m}$; $b \in \mathbb{R}^n$

Salida: $x \in \mathbb{R}^m$

1 $x \leftarrow$ Aproximación del problema mín $f^T x$, sujeto a $Ax \leq b \wedge x \geq 0$

2 Redondear la solución a dos decimales

A.2. Problema Maestro (*Master Problem*)

Este algoritmo aproxima la solución entera del Problema Maestro, el cual es de la siguiente forma

$$\begin{aligned} \text{minimizar} \quad & \sum_{j=1}^{m} x_j \\ \text{Sujeto a:} \quad & \sum_{j=1}^{m} a_{ij} x_j \geq d_i;\ i = 1, 2, ..., n \\ & x \in (\mathbb{N}^*)^m \end{aligned} \tag{A.2}$$

utilizando el algoritmo anterior `Aprox`.

Recuerde que x_j representa la cantidad de copias que se deben imprimir del patrón j, además, la cantidad de copias de cada gráfico debe ser mayor o igual a la demanda de cada uno de ellos. Así, si alguno de los elementos de x no es entero, el algoritmo lo redondeará al entero mayor. De esa manera se garantiza que la demanda se satisface y además que la solución obtenida es entera.

Algoritmo A.2: Aproximación al Problema Maestro

`[x] = AproxMast(A,d,m)`

Entrada: $A \in \mathbb{R}^{n \times m}$; $d \in \mathbb{R}^m$; $m \in \mathbb{R}$

Salida: $x \in \mathbb{R}^m$

1 $x \leftarrow$ Aproximación del problema mín $\sum_{j=1}^{m} x_i$, sujeto a $Ax \geq d \wedge x \geq 0$

2 **for** $i = 1 : m$ **do**

3 **if** $x_i \notin \mathbb{Z}$ **then**

4 $x_i \leftarrow$ Parte entera de $x_i + 1$

5 **end**

6 **end**

A.3. Subproblema (*Pricing Problem*)

Este algoritmo aproxima la solución entera de cada uno de los subproblemas

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{n} y_i a_{ij} \\ \text{Sujeto a:} \quad & \sum_{i=1}^{n} l_i a_{ij} \leq L \\ & A_{:j} \in (\mathbb{N})^n \end{aligned} \tag{A.3}$$

utilizando el algoritmo `Aprox`.

Recuerde que a_{ij} representa la cantidad de veces que se repite el gráfico i en el patrón j, además, las suma de las dimensiones de todos los gráficos presentes en un patrón debe ser menor o igual a la dimensión del molde. Por lo tanto, el alguna de las entradas de $A_{:j}$ no es entera se redondeará al entero menor. Así, la solución va a ser entera y satisfacerá la restricción.

Algoritmo A.3: Aproximación al Subproblema

`[a,z] = AproxPric(y,l,L,n)`

Entrada: $y \in \mathbb{R}^n$; $l \in \mathbb{R}^n$; $L \in \mathbb{R}$; $n \in \mathbb{R}$

Salida: $a \in \mathbb{R}^n$; $z \in \mathbb{R}$

1 $a \leftarrow$ Aproximación del problema máx $\sum_{i=1}^{n} y_i a_{ij}$, sujeto a $lA_{:j} \leq L \wedge a_{:j} \geq 0$

2 **for** $i = 1 : n$ **do**

3 **if** $a_{ij} \notin \mathbb{Z}$ **then**

4 $a_{ij} \leftarrow$ Parte entera de a_{ij}

5 **end**

6 **end**

7 $z \leftarrow$ Valor óptimo de la función objetivo