Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica

Rudy Alberto López Potosme*

* Licenciado en Ciencias de la Educación con mención en Matemática, Especialista en la enseñanza de Matemática en Educación Primaria. Máster en Formación de Formadores, Máster en Matemática Aplicada, por la Universidad Nacional Autónoma de Nicaragua. Docente Titular de la Universidad Nacional Autónoma de Nicaragua. Managua, Nicaragua.

RESUMEN

Los problemas de Optimización Combinatoria aparecen en diversos contextos, como la distribución de carga física o eléctrica, detección de patrones de corte de piezas, redes de tráfico o telecomunicaciones, horarios de transportes laborales y escolares, fabricación de circuitos electrónicos, secunciacion de actividades en una empresa, entre otros. Esto hace que dichos problemas sean actrativos para estudiar ya sea desde el punto de vista teórico o práctico.

El propósito fundamental del trabajo fue la construcción de una metodología para abordar los problemas de Optimización Combinatoria, particularmente los siguientes: el Problema de la Mochila, Problema de la Ruta más Corta, Problema de Corte de Piezas y el Problema del Agente Viajero. Cabe mencionar que esta metodología se caracteriza por resolver de una forma intuitiva, sencilla y práctica los problemas antes mencionados, utilizando algoritmos exactos y heurísticos, además de la implementación de la herramienta compuntacional WinQsb.

En lo que respecta al camino seguido para la realización del trabajo, es posible señalar que primeramente se realizó la construcción de una reseña histórica con la cual es posible comprender el génesis, desarrollo y el nivel de aplicación que tiene la Investigación de Operaciones, y en particular de los problemas de Optimización Combinatoria. Posteriormente para la elaboración de la metodología de solución que se propone, se revisó de forma exhaustiva el estado del arte de los modelos matemáticos, y de las diferentes técnicas de solución de los problemas en cuestión.

La forma en como resolver o tratar cada problema de Optimización Combinatoria se estructuró de la misma manera: a) Introducción, aquí se trata de dejar claro el problema, una breve reseña histórica, aplicaciones del problema y algunos algoritmos para resolvero; b) Formulación matemática del problema, se muestra la función objetivo y las restricciones del modelo; c) Tratamiento metodológico, se resuelven ciertos problemas mediante la implementación de diversos algoritmos; seguidamente se presentan d) Problemas propuestos; e) Problemas resueltos y f) Bibliografía, esta se decidió elaborarla para cada problema, dada la naturaleza heterogénea de los mismos.

Palabras claves: Investigación de Operaciones, Optimización Combinatoria, algoritmos exactos y heurísticos.

INTRODUCCIÓN

En lenguaje coloquial optimizar es equivalente a mejorar, y cuando hablamos de mejoras nos referimos a aquellas que se hacen evidente en los atributos de las situaciones, por ejemplo cuando vamos de compras al supermercado queremos optimizar nuestro dinero, lo que significa adquirir la mayor cantidad de productos a un menor precio. Este término simple y sencillo podemos aplicarlo a muchos entornos, por ejemplo, en el día a día, hablamos de optimizar el espacio dentro de algo, optimizar los costos, optimizar el trabajo que realizamos a diario; en actividades industriales, por ejemplo se habla de maximizar la producción, minimizar los costos, optimizar la energía, optimizar la fuerza de trabajo y el tiempo, etc. Por lo que en realidad cuando hablamos de optimización nos referimos al propósito de hacer más con menos.

Particularmente en el contexto matemático, optimizar es el proceso de tratar de encontrar la mejor solución posible para un problema determinado. Es decir que el objetivo de un problema de Optimización es encontrar la solución óptima dado un criterio para discriminar entre dos soluciones. Es decir, se trata de encontrar el valor de las variables de decisión (sujetas a restricciones) para los que una determinada función objetivo alcanza su valor máximo o mínimo.

Cabe señalar que los problemas de Optimización en los que las variables de decisión son enteras, es decir, donde el espacio de soluciones está formado por ordenaciones o subconjuntos de números naturales, reciben el nombre de Optimización Combinatoria. En este caso, se trata de hallar el mejor valor de entre un número finito o numerable de

soluciones viables. Sin embargo la enumeración de este conjunto resulta prácticamente intratable, aún para problemas de tamaño moderado.

Una gran cantidad y diversidad de problemas de optimización de recursos, ligados a situaciones reales, pueden ser formulados como problemas de Optimización Combinatoria: el ruteo y carga de vehículos en redes de distribución, el diseño de redes de telecomunicación, la planificación de la producción, la selección de carteras financieras, la asignación de tareas a procesadores, la asignación de tripulaciones en líneas aéreas, la planificación de la generación de electricidad y la distribución de ambulancias en una región para asegurar un cierto nivel de servicio a su población.

Es importante señalar que este trabajo versa sobre "Algunos problemas clásicos de Optimización Combinatoria: una propuesta metodológica". Con el desarrollo del mismo se tratan de resolver de una forma intuitiva, sencilla y práctica los siguientes problemas clásicos de Optimización Combinatoria: el Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Problema del Agente Viajero (Travelling Salesman Problem). Es decir que la idea fundamental radica en proponer una metodología para abordar los problemas de Optimización Combinatoria antes mencionados, cuando tenemos asosciados a ellos, instancias de baja dimensionalidad.

Cada uno de estos problemas de Optimización Combinatoria se abordan siguiendo un mismo esquema central: introducción, formulación matemática del problema, tratamiento metodológico, problemas propuestos y su solución correspondiente, finalmente se presenta las referencias bibliográficas. En la introducción se trata de dejar claro de qué trata el problema, se expone una breve reseña histórica, algunas aplicaciones del problema, además se comenta acerca de las diversas técnicas resolución; en la formulación matemática se explica cuál es la función objetivo y las restricciones correspondientes; en el tratamiento metodológico se resuelven algunos problemas de baja dimensionalidad mediante la implementación de diversos algoritmos; seguidamente se propone una colección de problemas y luego se resuleven con almenos uno de los algoritmos previamente ejecutado; en el caso de la bibliografía, esta se decidió elaborarla para cada problema, dada la naturaleza heterogénea de los mismos.

Uno de los aspectos principales para dar inicio a este trabajo de investigaicon, fue el hecho que al revisar los libros de Investigación de Operaciones a los cuales tanto docentes como estudiantes tienen acceso, fue evidenciar que la mayor parte de ellos abordan generalmente

los problemas de Programación Lineal y Problemas de Redes. Dejando a un lado los problemas de Optimización Combinatoria, que representan una gran parte de los problemas prácticos que se necesitan resolver y que cuya solución es demandada continuamente la sociedad, por las empresas que solicitan la resolución de problemas de optimización de recursos o mejoramiento del servicio.

Además de lo anterior se aprecia que los problemas son abordados con una metodología árida, descontextualizada, los problemas se resuelven de una solo forma o mediante un único algoritmo, y muy pocas veces o casi nunca se trabaja con algoritmos heurísticos, que son procedimientos que desempeñan un papel importante en la resolución de problemas de Optimización Combinatoria con la idea de encontrar buenas soluciones factibles de manera rápida, cuando se tienen problemas cuyas instancias son de una gran dimensión y por tanto es el tipo de problema predominante la vida cotidiana.

La realización de este trabajo se justifica porque los resultados del mismo están encaminados a realizar un aporte a la solución del problema de la escasa o prácticamente nula existencia de libros u otros tipos de materiales donde se aborden los principales problemas de Optimización Combinatoria utilizando una metodolología intuitiva, sencilla y práctica para abordar los problemas en mención.

Con los resultados del trabajo se generan ideas que podrían tener un impacto directo en el proceso de mejoramiento de los programas de asignaturas del curso Investigación de Operaciones, el cual es ofertado en las Falcultades Ingeniería, Ciencias Econónicas y Ciencias Exactas de la distintas universidades del país. Mejoramiento, en el sentido de fortalecer la formación en Matemática Aplicada de los futuros profesionales de las Facultades antes mencionadas.

El principal objetivo del trabajo fue elaborar una propuesta de metodología para el tratamiento de algunos problemas de Optimización Combinatoria: el Problema de la Mochila (Knapsack Problem), Problema de la Ruta más Corta (Shortest Path Problem), Problema de Corte de piezas (Cutting Stock Problem) y el Problema del Agente Viajero (Travelling Salesman Problem).

DESARROLLO

A continuación se muestra un resumen de la forma en como se abordaron dos, de los cuatro problemas de Optimización Combinatoria trabajados.

1. El Problema de la Mochila

1.1 Introducción

El Problema de la Mochila es un problema simple de entender: hay una persona que tiene una mochila con una cierta capacidad y tiene que elegir que elementos ubicará en ella. Cada uno de los elementos tiene un peso y aporta un beneficio. El objetivo de la persona es elegir los elementos que le permitan maximizar el beneficio sin excederse de la capacidad permitida.

A la vez es un problema complejo, si por complejidad nos referimos a la computacional. Un problema se cataloga como inherentemente difícil si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado. El Problema de la Mochila forma parte de una lista histórica de problemas NP-Completos elaborada por Richard Karp en 1972.

En el caso del Problema de la Mochila, si contáramos con 4 productos, para saber cuál es la mejor solución podríamos probar las $2^4 = 16$ posibilidades. El 2 se desprende del hecho de que cada decisión es incluir o no al producto y el 4 de la cantidad de productos. 16 posibilidades es un número manejable, sin embargo, si la cantidad de elementos por ejemplo ascendiera a 20, tendríamos que analizar nada más y nada menos que $2^{20} = 1\ 048\ 576$ posibilidades.

1.2 Formulación matemática del Problema de la Mochila

Supongamos que estamos planeando un viaje de senderismo; y estamos, por lo tanto, interesados en llenar una mochila con los elementos que se consideran necesarios para el viaje. Hay n diferentes tipos de elementos que se consideran deseables; estos podrían

incluir una botella de agua, manzana, naranja, sándwich, etc. adelante. Cada tipo de elemento tiene un conjunto dado de dos atributos, es decir, un peso (o volumen) y un valor que cuantifica el nivel de importancia asociado con cada unidad de ese tipo de elemento.

Una gran variedad de problemas de asignación de recursos se puede convertir en el marco de un Problema de la Mochila. La idea general es pensar en la capacidad de la mochila como la cantidad disponible de un recurso y los tipos de elementos como las actividades a las que este recurso puede ser asignado. Dos ejemplos rápidos son la asignación de un presupuesto de publicidad para las promociones del individuo productos y la asignación de su esfuerzo a la preparación de los exámenes finales en diferentes asignaturas.

Los datos del problema se pueden expresar en términos matemáticos de la siguiente manera:

- Los objetos están numerados por el índice i variando de 1 a n.
- Los números w_i y p_i representan el peso y el valor del número i.
- La capacidad de la mochila se denomina en esta fórmula W .

Existen muchas manera de llenar la mochila, para decidir a cada uno de ellos debemos de decir para cada objeto si lo metemos a la mochila o no, pudiendo utilizar el código binario que cuando $x_i = 1$, metemos el objeto a la mochila, o $x_i = 0$, se pone afuera, y para ir llenando esta mochila podemos utilizar un vector de contenido, que comprende: $X = (x_1, x_2, \ldots, x_n)$, entonces podemos expresar una función del contenido del vector.

Para un contenido dado X , el valor total de la bolsa es:

$$Z(x) = \sum_{i=1}^{n} x_i p_i$$

De la misma manera, la suma de los pesos de los objetos es:

$$W(x) = \sum_{i=1}^{n} x_i w_i$$

El problema entonces lo podemos enunciar como un contenido de vectores $X = (x_1, x_2, \ldots, x_n)$, que tienen componentes de ceros y unos, teniendo como máximo la restricción de la función $Z(x)$.

$$W(x) = \sum_{i=1}^{n} x_i w_i \leq W$$

Esto quiere decir que la suma de los pesos (o sea, la función $W(x)$), de los objetos que pusimos en la mochila no deben de superar la capacidad de esta, (o sea, la W).

Podemos decir que:

- $Z(x)$ es una función objetivo (como su nombre lo dice, representa el objetivo del problema, esta expresión se maximiza o se minimiza.
- Un vector X que cumple con la restricción W en la tercera fórmula se le nombra factible (o sea, que se pude hacer).
- Si tenemos un resultado máximo en $Z(x)$, entonces X es óptimo.
- Se le pueden agregar otras restricciones según tengamos un caso, en esta liga, encontramos diferentes casos singulares.

Con esto podemos argumentar que tenemos un problema de decisión cuando para decidir a cada uno de ellos debemos de decir para cada objeto si lo metemos a la mochila o no, que cuando $x_i = 1$, metemos el objeto a la mochila, o $x_i = 0$, se pone afuera y podemos decir que es un problema de optimización porque podemos encontrar funciones objetivo, valores óptimos utilizando las declaraciones matemáticas.

1.3 Tratamiento metodológico

El tratamiento metodológico que se realizará con el Problema de la Mochila es el siguiente: primero se resolverán ciertos problemas de baja dimensión, mediantes diferentes métodos; para iniciar se utilizará un algoritmo exacto (búsqueda exhaustiva), después un algoritmo de Programación Dinámica, seguidamente se propone encontrar la solución con un algoritmo heurístico y luego se confirmarán los resultados obtenidos con la aplicación de un herramienta computacional como lo es el WinQsb.

Ejemplo 1. Un turista nacional planea salir el fin de semana a la Isla de Ometepe. Hay cuatro artículos que desea llevar consigo, pero entre todos sobrepasan las 5 kilogramos que considera puede cargar. El peso y valor de cada artículo se muestran en la siguiente tabla

Artículo	**1**	**2**	**3**	**4**
Peso	2	3	4	5
Valor	3	4	5	6

¿Qué artículos tendría que llevar para que el valor de la mochila sea máximo?

Solución 1: (Implementando el algoritmo exacto de búsqueda exhaustiva)

La primera forma para resolver este problema será mediante la implementación del algoritmo de búsqueda exhaustiva. La búsqueda búsqueda exhaustiva, también es conocida búsqueda combinatoria, búsqueda exhaustiva o sencillamente fuerza bruta, es una técnica trivial, que consiste en enumerar sistemáticamente todos los posibles candidatos para la solución de un problema, con el objetivo de determinar si dicho candidato satisface la solución al mismo.

La búsqueda por fuerza bruta es sencilla de implementar y, siempre que exista, encuentra una solución. Sin embargo, su costo de ejecución es proporcional al número de soluciones candidatas, el cual es exponencialmente proporcional al tamaño del problema. Por el contrario, la búsqueda por fuerza bruta se usa habitualmente cuando el número de soluciones candidatas no es elevado.

Para resolver el problema primeramente se planteará el modelo matemático asociado a dicho problema:

Función Objetivo:

$$Max \quad Z = 3x_1 + 4x_2 + 5x_3 + 6x_4$$

Sujeto a:

$$2x_1 + 3x_2 + 4x_3 + 5x_4 \leq 5$$

Seguidamente, para lograr visualizar de una mejor manera la totalidad de posibles opciones de solución, se elabora un diagrama de árbol, el cual facilita la enumeración total de las soluciones posibles:

Al recorrer las ramas del árbol se va obteniendo el total de posibles soluciones al problema, las cuales las expresaremos en forma de un vector como el siguiente $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$. El valor 0 en una posición significa que ese objeto no se ingresará a la mochila, en cambio el valor 1 indica que el dicho objetos si será parte de la mochila. Por tanto, el vector antes expuesto indica que de los cuatro objetos posibles, a la mochila únicamente se meterá el artículo número 4.

Una vez encontradas y enumeradas todas las posibles soluciones, se procede a determinar con cuál de ellas se obtiene el beneficio máximo, respetando la restricción de volumen. A continuación se muestra una tabla donde se exponen las soluciones, el peso y beneficio a asociada a cada una de ellas.

Nº	Posible solución	Peso	Valor
1	$\begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$	0	0
2	$\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$	5	6
3	$\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}$	4	5
4	$\begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$	9	11
5	$\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$	3	4
6	$\begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$	8	10
7	$\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix}$	7	9
8	$\begin{bmatrix} 0 & 1 & 1 & 1 \end{bmatrix}$	12	15
9	$\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}$	2	3
10	$\begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix}$	7	9
11	$\begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}$	6	8
12	$\begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix}$	11	14
13	$\begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}$	5	7

Nº	Posible solución	Peso	Valor
14	$\begin{bmatrix}1 & 1 & 0 & 1\end{bmatrix}$	10	13
15	$\begin{bmatrix}1 & 1 & 1 & 0\end{bmatrix}$	9	12
16	$\begin{bmatrix}1 & 1 & 1 & 1\end{bmatrix}$	14	18

De todas las posibles opciones de respuestas, sólo algunas de ellas satisfacen (1, 2, 3, 5, 9 y 13) la restricción referida al peso de la mochila y de estas, la opción en la que se obtiene el valor máximo es la número trece $\begin{bmatrix}1 & 1 & 0 & 0\end{bmatrix}$, en la cual se incluyen en la mochila los artículos 1 y 2, obteniendo el valor óptimo de 7.

Solución 2: (Implementando un algoritmo exacto de Programación Dinámica)

La técnica de Programación Dinámica evita explorar todas las secuencias posibles por medio de la resolución de subproblemas de tamaño creciente y almacenamiento en una tabla de las soluciones óptimas de esos subproblemas para facilitar la solución de los problemas más grandes. A continuación se expone el algoritmo para resolver el problema propuesto.

Algoritmo Bottom up Approach

Para encontrar el valor máximo de la mochila:

para $i = 1$ hasta n
dp $[i][0] = 0$

para $j = 1$ hasta w
dp $[0]$ $[j] = 0$

para $i = 1$ hasta n
para $j = 1$ hasta w
si peso $[i] \leq W$
dp $[i]$ $[j] = max$ (dp $[i-1]$ $[j]$, dp $[i-1]$ $[j$-peso$[i]]$ + valor $[i]$)

de lo contrario
dp $[i]$ $[j]$ = dp $[i-1]$ $[j]$
Retornar dp $[n]$ $[w]$

Para determinar los objetos que serán parte de la mochila:

Mientras $i, j > 0$

si dp $[i]$ $[j] \neq$ dp $[i-1]$ $[j]$ (el i elemento es parte de la mochila)
y hacer $i = i - 1$, $j = j -$ peso$[i]$

de lo contrario
$i = i - 1$ (asume que el elemento i-ésimo no está en la mochila)

Mediante la implementación del algoritmo lo que se hace es rellenar la siguiente matriz

Nº	**Peso (Volumen)**	**Valor (Beneficio)**	**0**	**1**	**2**	**3**	**4**	**5**
0	Posición	0						
1	2	3						
2	3	4						
3	4	5						
4	5	6						

Como consecuencia de lo antes expuesto se obtiene la matriz que permite determinar el beneficio máximo que se alcanza con esta mochila, que en este caso corresponde al valor 7, y que es justamente el mismo valor generado mediante el algoritmo de fuerza bruta.

Nº	**Peso**	**Valor**	**0**	**1**	**2**	**3**	**4**	**5**
0	Posición	0	0	0	0	0	0	0
1	2	3	0	0	3	3	3	3
2	3	4	0	0	3	4	4	7
3	4	5	0	0	3	4	5	7
4	5	6	0	0	3	4	5	7

Es importante señalar que la tabla por sí sola, únicamente nos permite determinar la solución óptima al problema de la mochila, pero hasta este momento no se conocen los objetos que se introducirán en ella. Es justamente, la segunda parte del algoritmo quien nos permitirá solventar esta situación.

Esta segunda parte del algoritmo consiste en comparar el último valor de la matriz con el de la fila anterior (dp $[i]$ $[j] \neq$ dp $[i-1]$ $[j]$), si son distintos, el objeto i será parte de la

mochila y se calcula un nuevo i ($i = i - 1$), y un nuevo j ($j = j -$ peso[i]), si ocurre que son iguales, entonces se descarta el objeto i y se continua comparando, mientras $i, j > 0$.

Este procedimiento se muestra a continuación:

Como dp[4][5] = dp[3][5], entonces el objeto $i = 4$ se descarta, y lo mismo ocurre con los objetos $i = 3$, puesto que ocurre dp[3][5]= dp[2][5], ahora bien, como dp[2][5] ≠ dp[1][5], el objeto $i = 2$ será parte de la mochila y se procede a calcular un nuevo j, que en este caso sería $j = 5 - 3 = 2$, luego se continua con la comparación: dp[1][2] ≠ dp[0][2], esto hace indicar que el objeto $i = 1$ se deberá incluir en la mochila y hacemos $j = 2 - 2 = 0$. El algoritmo se detiene aquí puesto que debe ser $j > 0$, ante esta situación es posible afirmar que se está ante la presencia del valor óptimo [1 1 0 0], es decir que los objetos que se deben meter a la mochila son el 1 y 2, obteniendo de esta manera un valor máximo de 7.

Solución 3: (Implementando el algoritmo heurístico del coeficiente de rendimiento)

Una solución intuitiva pero que puede no ser óptima la podemos encontrar con un algoritmo que se implementa de la siguiente forma:

1. Ordenar la lista de objetos de forma descendente de acuerdo a la siguiente proporción

$$r_i = \frac{b_i}{v_i}$$

2. Seleccionar los objetos hasta llenar la mochila.
3. Ahora tenemos dos opciones: si el siguiente objeto ya no cabe completo en la mochila podemos quedarnos con una fracción de él, obteniendo un beneficio igual a

$$(pesototal - pesoactual)/pesoobjeto$$

o no meter ningún objeto más en la mochila. La primera opción se elige cuando el problema es de mochila fraccional y la segunda, cuando el problema es de mochila 0/1.

Al ejecutar el primer paso del algoritmo se obtiene la siguiente tabla:

Objeto	***b***	***v***	***r***
1	3	2	1.5
2	4	3	1.3
3	5	4	1.2
4	6	5	1.2

Mediante el segundo paso de la heurística mencionada, elegiríamos los productos 1 y 2, ya que al tratar de ingresar el objeto 3 se estaría excediendo la capacidad permitida. Entonces la solución sería [1 1 0 0], generando un beneficio de 7. Se puede notar que, en este caso la solución encontrada por el heurístico coincide con la solución óptima, encontrada primeramente cuando enumeramos todas las posibles opciones y luego al aplicar programación dinámica.

2. El Problema de Corte

2.1 Introducción

El Problema de Corte, al igual que otros problemas de Optimización Combinatoria, es fácil de definir intuitivamente, sin embargo no es fácil de modelar formalmente y presenta un alto grado de dificultad lo cual hace difícil el manejo computacional. En este problema se tiene un conjunto de piezas de diferentes tamaños y formas que deben ser localizadas sobre un tablero de material de mayor tamaño sin superponerse unas sobre otras. El objetivo de tal disposición es maximizar el área utilizada de forma que se generen la menor cantidad de área desperdiciada.

El Problema de Corte ha sido ampliamente estudiado en numerosas áreas de la industria y la investigación. Es un problema que se presenta en aquellas industrias en las que dentro de su proceso productivo es necesario cortar material que viene en dimensiones estándar, con el fin de obtener piezas en tamaños y formas requeridas, como por ejemplo: el corte de madera, de rollos de papel, de tela, de acero, entre otros. Para estas industrias es de gran importancia realizar este proceso de corte de una manera eficiente buscando minimizar el desperdicio y los demás costos asociados al proceso, teniendo en cuenta las restricciones técnicas y de demanda que el sistema en cuestión impone.

En el Problema de Corte, los elementos pequeños son una lista de piezas demandadas, mientras que los objetos de mayor tamaño se conocen como stock o materia prima. Los clientes requieren las piezas, y las industrias o fábricas deben producir el conjunto de piezas demandadas usando la materia prima disponible para satisfacer las necesidades de sus clientes.

El Problema de Corte tiene diversas aplicaciones en las industria papelera, industria maderera, del vidrio, textil, de pieles y calzado, del metal, como también en el diseño de circuitos integrados, en el paginado de periódicos y sin lugar a dudas en la distribución ya que su aplicación permite obtener patrones de ubicación de productos que maximiza espacio de camiones de diferentes tipos.

2.2 Formulación Matemática del Problema de Corte de Piezas

Dado un stock de barras de tamaño *L* (objetos grandes) que deben ser cortadas en piezas de tamaño l_i (pequeños ítems) para atender la demanda N_i, $i = 1, \dots, m$ de las piezas de las piezas l_i. Las demandas son atendidas decidiendo sobre los distintos patrones (modelos de cortes) de la barra de tamaño *L*. El objetivo es minimizar el número de los grandes objetos utilizados.

El j-ésimo patrón o modelo de corte es una manera de dividir la barra de tamaño L en piezas de tamaño l_i y x_j es el número de veces que se utiliza el j-ésimo patrón de corte.

En la formulación del Problema de Corte de una dimensión como un programa no entero, Gilmore y Gomory (1960), la matriz $\boldsymbol{A}$ del problema lineal tiene m filas y un gran número de columnas, uno por cada posible patrón de corte. Así cada vector $(a_1, a_2, \dots, a_m)$ de enteros no negativos satisfaciendo $l_1 a_1 + l_2 a_2 + \cdots + l_m a_m \leq L$ es una columna de la matriz.

Así, tenemos que el modelo matemático para el problema de corte unidimensional es:

$$\begin{aligned} Min \quad & \sum_j x_j \\ s.a \quad & \sum_j a_{ij} x_j \geq N_i, \quad i = 1,\ldots,m \\ & x_j \geq 0 \end{aligned}$$

Y en forma matricial

$$\begin{aligned} Min \quad & 1.\mathbf{x} \\ s.a \quad & \mathbf{Ax} \geq \mathbf{N} \\ & \mathbf{x} \geq 0 \end{aligned}$$

Donde,

$x = (x_1, x_2, \ldots, x_j)$ es un vector columna de las variables x_j y para cada columna de la matriz $\boldsymbol{A}$, donde x_j es el número de veces que el patrón j se utiliza.
$1 = (1, 1, \ldots, 1)$ es un vector fila de elementos todos iguales a uno.
$N = (N_1, \ldots, N_m)$ es un vector columna de las demandas N_i.
$\boldsymbol{A}$: Matriz de m filas, cuyas columnas de estructura $(a_1, a_2, \ldots, a_m)$ son los posibles patrones de corte.
a_i: es el número de veces que la pieza l_m $(i = 1, \ldots, m)$ aparece en el patrón de corte.

Considerando x_j y N_i enteros, entonces estaríamos ante un problema de programación lineal entera. En la práctica se resuelve el problema de programación lineal asociado y redondeando la solución de este, podemos tener una solución satisfactoria para el problema de programación entera.

2.3 Tratamiento metodológico

Para el caso del Problema de Corte, el tratamiento metodológico consistirá en resolver problemas que posen un instancia de baja dimensión, para esto se implementará diferentes métodos de solución. Inicialemente se utilizará la técnica de Programación Lineal Entera, y para la cual usaremos el algoritmo Branch and Bound, auxiliándonos del WinQsb. Posteriormente se aborda el problema de cortar un tablero (bidimensional) en piezas más pequeñas, de tal manera que se optimice el beneficio. Para resolver dicho problema se implementará un algoritmo heurístico.

Ejemplo 1.

Supóngase que se tiene varillas de acero, en cantidad suficiente, de 12 metros de longitud cada una, y que se requieren cortes de las siguientes dimensiones y cantidades:

Dimensiones (m)	Cantidad
3	100
5	150
7	200
8	85

Tabla X. datos del ejemplo 1

Se desea saber cómo cortar las varillas de 12 metros, de tal forma que el desperdicio sea mínimo. Se considera, en este caso, el desperdicio como cortes menores que 3 m de longitud.

Solución: (Aplicando Programación Lineal Entera con WinQsb)

Podemos deducir que existen diversas formas posibles y lógicas de cortar cada varilla de 12 *m*, de tal forma que se obtengan las varillas de las dimensiones deseadas. Para poder formular el modelo deben describirse exhaustivamente todas estas formas. Para este caso, las combinaciones posibles o los patrones de corte factibles son los siguientes:

	Patrón de corte					
Dimensión de la varilla	**1**	**2**	**3**	**4**	**5**	**6**
3	4	2	1	1	0	0
5	0	1	0	0	2	1
7	0	0	1	0	0	1
8	0	0	0	1	0	0
Desperdicio (D)	0	1	2	1	2	0

Tabla X. Posibles cortes en el ejemplo 1

Es decir que existen 6 formas lógicas de cortar las varillas de 12 m. Por ejemplo, el patrón de corte número 3 se obtiene una varilla de 3 m y una varilla de 7 m, produciéndose un desperdicio de 2 m. Los demás patrones de corte de interpretan de forma análoga.

Con base a los 6 patrones de corte identificados en la solución 1, podemos definir las variables de decisión, de la siguiente manera:

Sea x_i el número de varillas de 12 m a cortar según el patrón $i = 1, 2, \dots, 6$

De manera que el modelo de Programación Lineal Entera será el siguiente:

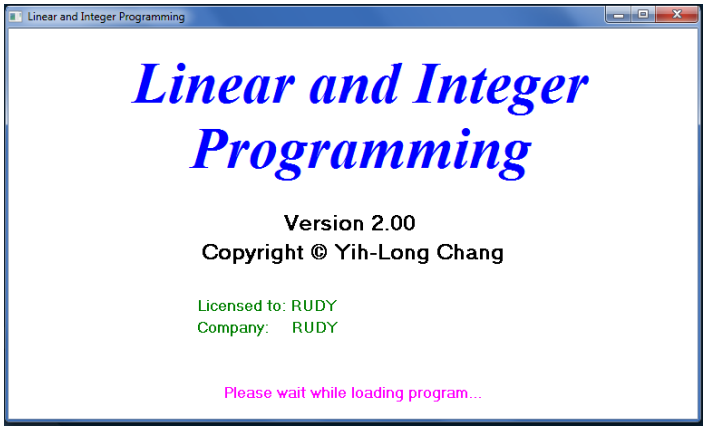
$$Minimizar\ D = x_2 + 2x_3 + x_4 + 2x_5$$

$$Sujeto\ a:$$

$$\begin{aligned} 4x_1 + 2x_2 + x_3 + x_4 &\geq 100 \\ x_2 + 2x_5 + x_6 &\geq 150 \\ x_3 + x_6 &\geq 200 \\ x_4 &\geq 85 \\ x_i \geq 0, i = 1, 2, 3, 4, 5, 6 & \end{aligned}$$

Una vez construido el modelo procederemos a resolverlo mediante Branch and Bound.

a) Resolviendo el problema relajado (sin considerar que las variables deben ser enteras) con el Winqsb:



b) Ingresamos el modelo relajado

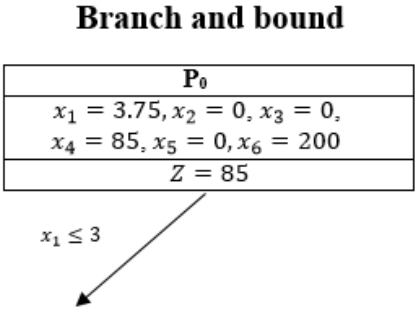
Variable -->	X1	X2	X3	X4	X5	X6	Direction	R. H. S.
Minimize	0	1	2	1	2	0		
C1	4	2	1	1	0	0	>=	100
C2	0	1	0	0	2	1	>=	150
C3	0	0	1	0	0	1	>=	200
C4	0	0	0	1	0	0	>=	85
LowerBound	0	0	0	0	0	0		
UpperBound	M	M	M	M	M	M		
VariableType	ontinuou	ontinuou	Continuou:	ontinuou	Continuou:	Continuou:		

c) Obtenemos la solución:

	Decision Variable	Solution Value	Unit Cost or Profit c[j]	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c[j]	Allowable Max. c[j]
1	X1	3.7500	0	0	0	basic	0	2.0000
2	X2	0	1.0000	0	1.0000	at bound	0	M
3	X3	0	2.0000	0	2.0000	at bound	0	M
4	X4	85.0000	1.0000	85.0000	0	basic	0	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	2.0000
	Objective	Function	(Min.) =	85.0000	(Note:	Alternate	Solution	Exists!!)

Por tanto obtenemos nuestra primera solución al problema (3.75, 0, 0, 85, 0, 200) y Z = 85. Puesto que las soluciones no son enteras procedemos a ramificar tomando las variables cuyo valor no es entero, en este caso x_1.

Luego escogemos la variable x_1 y agregamos una rama en donde $x_1 \leq 3$, es decir la parte entera de 3.75



Por lo tanto nuestro problema ahora queda de la siguiente forma:

$Minimizar\ D = x_2 + 2x_3 + x_4 + 2x_5$

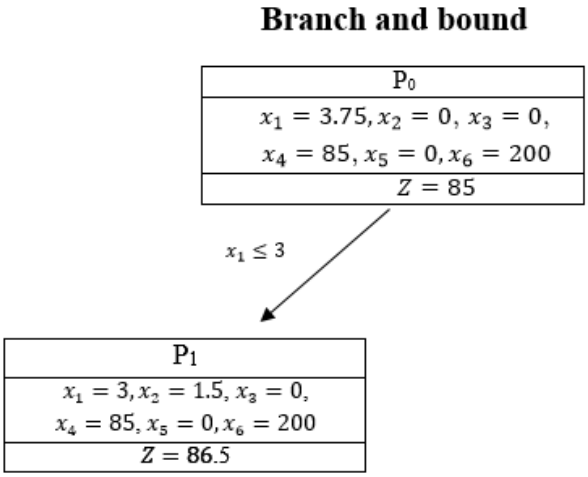
$Sujeto\ a$:

$$\begin{aligned} 4x_1 + 2x_2 + x_3 + x_4 &\geq 100 \\ x_2 + 2x_5 + x_6 &\geq 150 \\ x_3 + x_6 &\geq 200 \\ x_4 &\geq 85 \\ x_1 &\leq 3 \\ x_i &\geq 0 \end{aligned}$$

Resolviendo este problema mediante el WinQsb, tenemos:

	Decision Variable	Solution Value	Unit Cost or Profit c[j]	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c[j]	Allowable Max. c[j]
1	X1	3.0000	0	0	0	basic	-M	2.0000
2	X2	1.5000	1.0000	1.5000	0	basic	0	2.0000
3	X3	0	2.0000	0	1.5000	at bound	0.5000	M
4	X4	85.0000	1.0000	85.0000	0	basic	0.5000	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	1.5000
	Objective	Function	(Min.) =	86.5000	(Note:	Alternate	Solution	Exists!!)

Es decir que la solución para dicho modelo es (3, 1.5, 0, 85, 0, 200) y Z = 86.5. Luego procedemos a generar la rama que corresponde:



Como se puede apreciar, tenemos una solución con variables no enteras y que además generan un valor de la función objetivo superior a la solución del problema inicial. Posteriormente, procedemos crear una rama al lado derecho y agregamos la restricción $x_1 \geq 4$. Y en este caso obtenemos el problema inicial más esta última restricción:

$$Minimizar\ D = x_2 + 2x_3 + x_4 + 2x_5$$

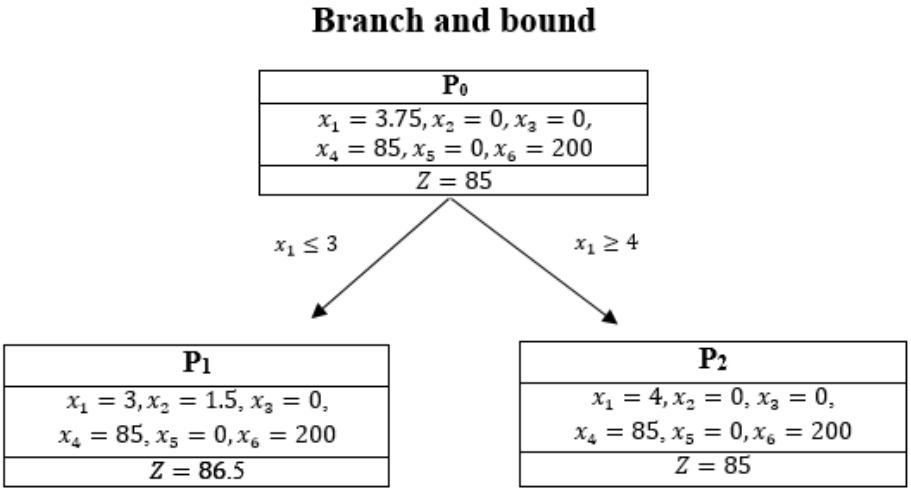
$$Sujeto\ a:$$

$$4x_1 + 2x_2 + x_3 + x_4 \geq 100$$
$$x_2 + 2x_5 + x_6 \geq 150$$
$$x_3 + x_6 \geq 200$$
$$x_4 \geq 85$$
$$x_1 \geq 4$$
$$x_i \geq 0$$

Seguidamente resolvemos el modelo y tenemos la siguiente tabla con la solución

	Decision Variable	Solution Value	Unit Cost or Profit c[j]	Total Contribution	Reduced Cost	Basis Status	Allowable Min. c[j]	Allowable Max. c[j]
1	X1	4.0000	0	0	0	basic	0	M
2	X2	0	1.0000	0	1.0000	at bound	0	M
3	X3	0	2.0000	0	2.0000	at bound	0	M
4	X4	85.0000	1.0000	85.0000	0	basic	0	M
5	X5	0	2.0000	0	2.0000	at bound	0	M
6	X6	200.0000	0	0	0	basic	0	2.0000
	Objective	Function	(Min.) =	85.0000	(Note:	Alternate	Solution	Exists!!)

En este caso obtenemos como solución (4, 0, 0, 85, 0, 200) y Z = 85. Creando la rama que corresponde tendríamos la gráfica:



En la rama del lado derecho nos encontramos con una solución entera, puesto que el valor de las variables de decisión es un número entero, y dado que en la rama izquierda nos encontramos con una solución no entera y que el valor de la función objetivo es superior a la que encontramos en el lado derecho, entonces podemos decir que ya hemos encontrado la solución óptima del problema (4, 0, 0, 85, 0, 200) y con ella tenemos un valor óptimo de 85.

CONCLUSIONES

- Se constuyó una reseña histórica por medio de la cual es posible entender aspectos relacionados con el génesis, desarrollo y el nivel de aplicación que tiene actualmente la Investigación de Operaciones, y en particular de los problemas de Optimización Combinatoria.

- Se realizó una revisión del estado del arte de las técnicas utilizadas para resolver los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y Travelling Salesman Problem.

- Se planteó la formulación matemática de los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y Travelling Salesman Problem.

- Se implementaron algoritmos exactos y heurísticos (Búsqueda exahaustiva, Programación Dinámica, Programación Lineal Entera, Branch and bound, Dijkstra, Floyd-Warshall, Bellman-Ford, heurísticos del coeficiente de rendimiento, heurístico de corte, el heurístico del vencino más cercano) para tratar los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y Travelling Salesman Problem.

- Se implementó la herramienta computacional (WinQsb) para resolver los problemas de Optimización Combintaria: Knapsack Problem, Shortest Path Problem, Cutting Stock Problem y Travelling Salesman Problem.

- Se diseñó una metodología para tratar los problemas Optimización Combinatoria, dicha metodología dista de la forma en que normalmente los libros clásicos tratan estos temas, ya que los problemas se resuelven de una forma intuitiva, sencilla y práctica, la cual

fue estructurada de la siguiente manera: a) Introducción; b) Formulación matemática del problema; c) Tratamiento metodológico; d) Problemas propuestos; e) Problemas resueltos y f) Bibliografía.

PERSPECTIVA DE FUTURO

- Implementar la metodología propuesta para abordar otros problemas de Optimización Combinatoria: Problema de Cubrimiento Máximo, Problema de Coloreo de Grafos, Problema de Transporte, Problema de Flujo de Costo Mínimo y Problema de Flujo Máximo.

BIBLIOGRAFÍA

- Álvarez-Valdés Ramón, Parajón Antonio, y Tamarit José Manuel. (2007). *GRASP and Path Relinking for the Two-Dimensional Two-Stage Cutting-Stock Problem*. INFORMS Journal on Computing.

- Álvarez-Valdés Ramón, Parajón Antonio, y Tamarit José Manuel. (2002) "A Tabu Search Algorithm for Large-scale Guillotine (Un) Constrained Two-dimensional Cutting Problems", Computers & Operations Research 29, 925-947.

- Bazaraa, S., Javis, John J. (1996). Programación lineal y flujo de redes. México: Limusa. Quinta edición.

- Bellman R., (1957). Dynamic Programming. Princeton University Press, Princeton, NJ.

- Dorta, I., León, C., Rodríguez, C., Rodríguez, G., y Rojas, A. (2003). Complejidad Algorítmica: de la Teoría a la Práctica. III Jornadas de Enseñanza Universitaria de Informática.

- Fayard, D. and Plateau G. (1994). An exact algorithm for the O-l collapsing knapsack problem. Discrete Applied Mathematics.

- Farley, A. (1990). The cutting stock problem in the canvas industry. European Journal of Operational Research, 44, 247-255.
- Gilmore, P., Gomory, R. (1961). A linear programming approach to the cutting stock problem. Operation Research. 9 849–859.
- Gilmore, P.C., & Gomory, R.E., (1966). The theory and computation of knapsack functions, Operations Research 14, 1045-1074.
- Kantorovich, L., Zalgaller V (1951). Calculation of Rational Cutting of Stock. Lenizdat, Leningrad.
- Martello, S., Toth, P. (1990). Knapsack Problems. Great Britain: John Wiley & Sons.
- N. Martí Oliet, Y. Ortega Mallén, A. Verdejo. (2013). Estructuras de datos y métodos algorítmicos. España: Garceta.
- Oliveira, J., Ferreira, J. (1990). An improved version an improved version of Wang's algorithm for two - dimensional Cutting Problems. EJOR 44. pp. 256-266.
- Pisinger, D. (2003). Where are the hard knapsack problems?. Technical Report 2003/8, DIKU, University of Copenhagen, Denmark.
- Velasco, J. (2010). NP-Completeness: Complejidad del problema de la Mochila. Presentación de Tesis. Facultad de Ingeniería Mecánica y Eléctrica. División de Posgrado en Ingeniería en Sistemas. Universidad Autónoma de Nuevo León.