VNIVERSIDAD NACIONAL AVFNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

FACULTAD DE INGENIERÍA

PROPUESTA DE UN ALGORITMO *BRANCH AND CUT* PARA RESOLVER EL PROBLEMA DE DISTRIBUCIÓN DE PLANTA BIDIMENSIONAL CON ÁREAS DESIGUALES

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN INGENIERÍA

INGENIERÍA DE SISTEMAS - INVESTIGACIÓN DE OPERACIONES

P R E S E N T A:

ING. LAURA BERENICE GUERRERO AYALA

TUTOR:

Dra. Mayra Elizondo Cortés

Ciudad Universitaria Noviembre 2010

Jurado asignado:

Presidente:	Dr. Aceves García Ricardo
Secretario:	Dra. Sánchez Larios Herica
1^{er} Vocal:	Dra. Elizondo Cortés Mayra
1^{er} Suplente:	Dra. Flores De la Mota Idalia
2^{o} Suplente:	Dr. Estrada Medina Juan Manuel

Lugar donde se realizó la tesis:

Departamento de Ingeniería de Sistemas
Laboratorio de Investigación de Operaciones (IDEO)
de la Universidad Nacional Autónoma de México (UNAM).

Tutor de tesis:

Dra. Mayra Elizondo Cortés.

Agradecimientos

A Gerardo por su paciencia y colaboración en la realización de este trabajo de tesis, por su constante motivación, por estar siempre a mi lado apoyandome.

A la Dra. Mayra Elizondo Cortés, por su dedicación y ayuda en la dirección de este trabajo de tesis, pero sobretodo por sus enseñanzas y consejos.

A mis amigas, Adela, Fabiola e Iris, por su cariño y motivación en los últimos años.

A mi familia por su comprensión.

A los Doctores del área de Investigación de Operaciones por el valioso aprendizaje que recibí de ellos a lo largo de mi estancia en la maestría.

Agradezco especialmente su valioso apoyo a Ma. Guadalupe Rendón.

A la UNAM por haberme abierto las puertas y contribuir de forma muy positiva en mi creciemiento personal y profesional.

Al CONACyT, por haberme otorgado la beca que me permitió realizar mis estudios de maestría.

Al COMECyT, porque su apoyo me permitió concluir este trabajo de tesis.

Índice general

Índice de figuras

Índice de tablas

Resumen

El problema de distribución de planta consiste en determinar una disposición de las instalaciones de trabajo de forma tal que éstas no se superpongan, considerando ciertas restricciones de espacio para cada departamento, al mismo tiempo que se minimiza el costo del flujo de materiales entre instalaciones.

Este problema es del tipo combinatorio, por lo que el número de restricciones crece exponencialmente conforme aumenta el número de instalaciones involucradas y además pertenece a la clase NP-completo.

En el trabajo de investigación se propone el diseño de un algoritmo Branch and Cut (B & C) para resolver el problema de distribución de planta bidimensional con áreas desiguales, ya que en general el algoritmo permite manejar grandes cantidades de variables y restricciones; además el algoritmo B & C ha sido adaptado a bastantes y diversas aplicaciones, las cuales reportan buenos resultados.

El algoritmo B & C combina dos métodos exactos, Branch & Bound y Planos de Corte, ambos para solucionar problemas de optimización y combinatoria. El esquema general de solución es generar cortes globalmente válidos, tanto para los nodos del árbol de búsqueda, como para el problema original, y así obtener una secuencia de aproximaciones a la cubierta convexa del programa entero, esto es, ajustarse cada vez mejor al conjunto de soluciones factibles.

Se parte de una formulación matemática manejable para el problema, se considera el modelo propuesto por Heragu (2006), el cual es una linealización del ABSMODEL 3 de Kusiak y S.S. (1987).

El conjunto de restricciones que dificultan el problema son las que impiden el empalme de instalaciones. Se generan cortes válidos para el problema con el algoritmo entero-mixto de Gomory. Cuando dichos cortes no mejoran significativamente, el valor de la función objetivo entonces, se procede a ramificar sobre las variables binarias que involucra el modelo.

Hacer funcionar eficientemente todo el mecanismo no es sencillo, dado que constantemente se añaden restricciones, de tal forma conviene eliminar periódicamente aquellas que han dejado de ser útiles en las últimas iteraciones.

Se propone utilizar el algoritmo completo en ejemplos que involucran 2, 3 y 5 instalaciones para mostrar su funcionamiento.

Abstract

In the Facility Layout Problem *(FLP)* we determine the work stations placement in order to avoid overlapping, considering space restrictions for each department, and also minimizing the handling material cost between facilities. This is a combinatorial problem, where the number of restrictions grows exponentially as the number of departments involved, besides it belongs to the NP-complete class.

This research proposes a *Branch & Cut* algorithm design in order to solve the bi-dimensional Facility Layout Problem with unequal areas, since the algorithm allows manage several variables and restrictions; also, the *Branch & Cut* has been adapted for several and different applications, which have been reported good results.

The algorithm combines two exact methods, *B & B* and Cutting Planes, both are used for solve optimization and combinatorial problems. The general solution scheme is to generate globally valid cutting planes for the search tree nodes and for the original problem, such that we can obtain a sequence of approaches to the convex hull of the integer problem that is, tightening better the feasible solution set.

We start with a mathematical formulation easy to use, we consider the model proposed by Heragu (2006), which it's a linearization of the ABSMODEL 3 of Kusiak y S.S. (1987).

The restrictions set that makes difficult the problem, is the one that avoids overlapping among facilities. We generate valid inequalities with the Gomory's integer-mixed algorithm. When the cutting planes does not improve the objective function value significantly, we branch over the binary variables that the model has.

Making work this procedure efficiently is not simple, because we add more restrictions each time, so it is convenient to eliminate periodically those restrictions which have left to be useful in the last iterations.

We propose the use of the complete algorithm in examples with 2, 3 and 5 facilities.

Introducción

I. Planteamiento del problema

Las empresas manufactureras y de servicios invierten una gran cantidad de tiempo y dinero en diseñar o bien, rediseñar sus instalaciones. Esta es una problemática muy importante que debe ser tratada antes de que los productos sean elaborados o bien, los servicios sean prestados. Un diseño de instalaciones pobre puede traer resultados costosos y productos de baja calidad, bajo estado de ánimo por parte de los trabajadores e insatisfacción por parte del cliente. (Heragu, 2006)

El problema de distribución de planta consiste en determinar una disposición de las instalaciones de trabajo de forma tal que éstas no se superpongan, considerando ciertas restricciones de espacio para cada departamento, al mismo tiempo que se minimiza el costo del flujo de materiales entre instalaciones.

Los costos de flujo de materiales entre instalaciones van del 30 % al 75 % de los costos totales de manufactura. Con base en las características y demanda del producto se puede definir el grado de flexibilidad y ritmo de producción de la planta. (Dorf y A., 1994) Actualmente el ritmo de producción de las empresas requiere de Sistemas de Manufactura Flexible *(FMS)*, en donde es necesario decidir sobre la distribución de planta, porque las especificaciones del diseño y las necesidades cambian continuamente, lo cual ocasiona que los diseños iniciales de un *FMS* varíen.

Por lo tanto, existen fuertes razones que hacen necesaria una solución correcta del problema de distribución de planta, ya que una solución óptima permitirá el incremento de la productividad y la eficiencia de una empresa, al igual que una disminución de los movimientos de material, pérdidas económicas y riesgos de trabajo.

Así pues, el problema descrito puede plantearse como un problema de optimización, ya que se desea elegir el mejor acomodo de instalaciones dentro de un conjunto de posibles arreglos, es decir, tenemos un problema de optimización del tipo combinatorio.

Para encontrar la solución óptima se recurre a los algotimos exactos, ya que éstos aseguran obtener el mejor valor de la función objetivo.

Se realiza una aportación de carácter teórico, que implica el diseño de un algoritmo de tipo exacto para resolver el problema de distribución de planta, lo que a su vez podría llevar a un incremento en el número de instalaciones involucradas con una solución factible.

Para algunos problemas de Optimización Matemática existen algoritmos exactos mejores y más rapidos que algunas técnicas heúrísticas y metaheurísticas, respecto a la búsqueda del óptimo, la implementación del algoritmo Branch and Cut (B & C) ha resultado eficiente en problemas de programación entera mixta tales como: Planeación de tareas (*scheduling*), (Stecco *et al.*, 2008) y (Bixby y Lee, 1998); Cadena de Suministro (*supply chain*), (Perron *et al.*, 2010); Localización de Servicios (*facility location problem*), (Caprara y J.J, 1996) y el Agente Viajero (*traveling salesman problem*), (Padberg y G., 1991).

II. Objetivos

General:

- Diseñar un algoritmo Branch & Cut (B & C) para resolver el problema de distribución de planta bidimensional con áreas desiguales

Particulares:

- Contar con una formulación matemática manejable para el problema.
- Identificar el conjunto de restricciones que dificultan el problema.
- Generar cortes válidos para el problema a lo largo de todo el árbol de búsqueda.
- Desarrollar el algoritmo B & C propuesto en un ejemplo de tamaño pequeño que involucre hasta 5 instalaciones.
- Mostrar que se reduce el número de iteraciones que se obtienen con el algoritmo Branch & Bound (B & B) , para los ejemplos aquí desarrollados.

III. Estructura de la Tesis

El presente trabajo está organizado en cuatro capítulos. En el Capítulo 1, se da una introducción al problema de Distribución de Planta así como su clasificación y las diversas formulaciones matemáticas.

En el Capítulo 2, se introduce a cada uno de los métodos exactos de solución de los que se vale el algoritmo B & C. Se presenta el estado del arte de los métodos de solución al problema de Distribución de Planta y finalmente se explica de forma general el funcionamiento del algoritmo B & C.

El Capítulo 3 trata de la definción del modelo matemático del problema y el análisis de su estructura para la aplicación del algoritmo propuesto.

En el Capítulo 4 se desarrollan los ejemplos de aplicación del algoritmo, se muestra detalladamente el funcionamiento de éste para un ejemplo con dos instalaciones a ubicar; para el caso de los ejemplos que involucran tres y cinco departamentos se presentan los resultados obtenidos con algunos comentarios.

Finalmente se presenta un apartado de las conclusiones y extensiones del trabajo de investigación.

Capítulo 1

El problema de Distribución de Planta

1.1. Definición del problema de Distribución de Planta (FLP)

La distribución de planta es un concepto relacionado con la disposición de las máquinas, departamentos, estaciones de trabajo, áreas de almacenamiento, pasillos, y espacios comunes dentro de una instalación productiva propuesta o ya existente.

Dicho problema (*Facility Layout Problem*, y por sus siglas en inglés *FLP*), afecta de forma directa a la productividad de los sistemas de manufactura, por lo que se busca la ordenación óptima de los elementos de un sistema de producción.

El arreglo óptimo contempla aspectos geométricos y económicos. El punto de vista económico tiene que ver con el proceso productivo de la planta, se considera una medida de interacción entre instalaciones, de movimiento de material, capacidades de producción, etc.; por lo tanto, instalaciones con mayor interacción, o bien flujo, entre ellas deberán estar más cercanas. El aspecto geométrico se relaciona con la arquitectura del sistema, las dimensiones de una máquina o departamento así como su localización respecto a los demás.

Cada par de instalaciones tendrá asociada una distancia y un flujo entre ellas; durante la distribución de la planta, la cantidad de configuraciones que pueden asumir los departamentos es exponencial y la interacción entre ellos cambia, por lo que se busca arreglar los departamentos de forma tal, que se reduzca la circulación de información y/o material.

De la consideración de los aspectos anteriores se derivan diferentes formulaciones del problema, según el modelo geométrico adoptado para representar la solución y de acuerdo a la función objetivo a optimizar, que puede incluir términos cuantitativos como costos de instalación y de operación, además de términos cualitativos, como la preferencia de producción de ciertos artículos y aspectos arquitectónicos.

Se utilizarán de forma indistinta los términos planta, departamento o bien instalación, éstos se referirán a la unidad de trabajo a localizar, de igual forma ubicación, localidad y localización.

1.2. Clásificación de los problemas de Distribución de Planta.

En la literatura, Heragu (2006) y Solimanpur y Jafari (2008) han clasificado estos problemas como:

- **Unidimensional**: se refiere al arreglo de instalaciones a lo largo de una línea, es decir, distribuir departamentos a lo largo de un corredor, lo cual se puede reconocer en supermercados, hospitales, edificios de oficinas, libros en un estante, etc.
- **Bidimensional** : cuando la disposición de instalaciones puede variar en las direcciones horizontal y vertical.
- **Áreas iguales**: Cada instalación es de igual forma rectangular.
- **Área desiguales**: Cada instalación tiene una forma rectangular diferente y por consiguiente diferente área. Se tiene una mayor aplicación práctica en estos casos. Generalmente se aplica a los casos de manufactura celular, en donde se pretende una gran flexibilidad para producir diversos productos de demanda baja y continuar con la alta producción a gran escala.

1.3. Formulaciones Matemáticas

Considerando la clasificación anterior, se han realizado diferentes formulaciones matemáticas para el problema de Distribución de Planta (Kusiak y S.S., 1987), a acontinuación se presentan algunas de éstas.

1.3.1. Problema de Asignación Cuadrática (QAP)

Koopmans y Beckmann (1957) fueron pioneros en modelar el problema de localizar n departamentos/plantas con flujo de materiales entre ellos en n localizaciones, aunque en

esta formulación, no se considera la dimensión y forma de las instalaciones. Trabaja bajo el supuesto de que los departamentos tienen áreas iguales.

El nombre de *Quadratic Assignment Problem (QAP)* se debe a que se tiene una función objetivo de segundo orden, es decir, un producto entre dos variables. Se consideran las siguientes variables y parámetros:

n: número total de instalaciones y ubicaciones;

a_{ij}: ingreso neto de operar la instalación i en la localización j;

f_{ik}: flujo de materiales entre el departamento i y el departamento k;

c_{jl}: costo de transportar una unidad de material de la ubicación j a la ubicación l;

$$x_{ij} = \begin{cases} 1 & \text{si el departamento } i \text{ es asignado al departamento } j \\ 0 & \text{cualquier otro caso} \end{cases}$$

Se asume que,

a_{ij}: es el ingreso bruto menos el costo de la inversión;

f_{ik} es independiente de la localización de las instalaciones;

c_{jl} es independiente de las instalaciones, además de que será más barato transportar el material directamente de la instalación i a la instalación k, que realizar una escala en una tercera instalación. El modelo queda de la siguiente forma,

$$Maximizar \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{\substack{k=1 \\ i\neq k}}^{n}\sum_{\substack{i=1 \\ j\neq l}}^{n} f_{ik}c_{jl}x_{ij}x_{kl} \tag{1.1}$$

sujeto a:

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1, 2, 3, ..., n \tag{1.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad j = 1, 2, 3, ..., n \tag{1.3}$$

$$x_{ij} \in [0, 1] \tag{1.4}$$

$$i, j, k, l = 1, 2, 3, ..., n$$

La restricción 1.2 asegura la asignación de un solo departamento a una localidad, junto con la restricción 1.4 que certifica que no se asignen partes de departamentos a varias localidades. La restricción 1.3 garantiza que cada ubicación tenga asignado exactamente un solo departamento. Finalmente 1.4 define el intervalo de valores para las variables de decisión.

De dicha formulación se desprenden dos variantes del problema:

- Problema de Asignación Lineal *(Linear assignment problem - LAP)*: no se consideran flujos entre plantas, por lo que el problema se reduce a minimizar el costo de localización de instalaciones, las restricciones quedan de la misma forma.

- Problema del Agente Viajero *(Traveling salesman problem - TSP)*: a partir de las consideraciones del LAP, se observa que las restricciones 1.2 - 1.4 implican una representación matricial en la cual cada columna y cada fila tienen entradas 0, 1, es decir, se tiene la matriz unimodular, característica del problema del agente viajero.

Sin embargo, si a_{ij} fuera el costo de localizar y operar la planta i en la localidad j en lugar del ingreso neto, el modelo se puede reescribir como se muestra a continuación,

$$Minimizar \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{i=1}^{n} f_{ik}c_{jl}x_{ij}x_{kl}$$

Tomando en cuenta las restricciones anteriores 1.2 - 1.4.

Si se considera el problema de distribuir máquinas no se puede utilizar el modelo anterior, ya que no todas las máquinas tendrán las mismas dimensiones (área), entonces la distancia entre dos ubicaciones no puede ser determinada *a priori*. Las localizaciones no se conocen debido a que no se sabe cómo será el arreglo de las máquinas. La distancia d_{ij} entre la localización i y j depende de cada arreglo que se vaya realizando. Esto no sucede en problemas donde se tienen instalaciones de igual tamaño porque las localizaciones serán todas de las mismas dimensiones y la distancia no cambiará en cada arreglo.

Es posible modelar áreas desiguales de departamentos con la formulación QAP rompiendo los departamentos en pequeñas cuadrículas con iguales áreas y no permitir la separación de la cuadrícula que pertenece al mismo departamento asignando flujos artificiales entre las cuadrículas.

1.3.2. Problema de Cobertura Cuadrática (*Quadratic Set Covering Problem* - QSP)

Formulación que considera a d_{ij} variable en cada arreglo de instalaciones. Aquí, el total del área ocupada por las instalaciones se divide en un número de bloques; además, se considera al problema como bidimensional y de áreas desiguales (Bazaraa, 1975). Se utiliza la siguiente notación:

q: número de bloques en los que se divide el total del área ocupada por todas las instalaciones;

$I(i)$: número de localizaciones potenciales para la instalación i;

$d(j_i, l_k)$: distancia entre los centroides de las localizaciones j y l si la instalación i es asignada a la localidad j y la instalación k es asignada a la localidad l;

$$x_{ij} = \begin{cases} 1 & \text{si la instalación } i \text{ es asignada a la localización } j \\ 0 & \text{cualquier otro caso} \end{cases}$$

$$p_{ijt} = \begin{cases} 1 & \text{si el bloque } t \in J_i(j) \\ 0 & \text{cualquier otro caso} \end{cases}$$

El modelo se muestra a continuación:

$$Minimizar \quad \sum_{i=1}^{n}\sum_{j=1}^{I(i)} a_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{I(i)}\sum_{k=1}^{n}\sum_{i=1}^{I(k)} f_{ik}d(j_i, l_k)\, x_{ij}x_{kl} \tag{1.5}$$

sujeto a:

$$\sum_{j=1}^{I(i)} x_{ij} = 1; \qquad i = 1, 2, 3, ..., n \tag{1.6}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{I(i)} p_{ijt}x_{ij} \leq 1; \qquad t = 1, 2, 3, ..., q \tag{1.7}$$

$$x_{ij}, x_{kl} \in [0, 1] \tag{1.8}$$

$$\begin{aligned} i, k &= 1, 2, 3, ..., n \\ j, l &= 1, 2, 3..., I(i) \end{aligned}$$

En donde 1.6 asegura que cada instalación sea asignada a exactamente una localización, 1.7 indica que cada bloque es ocupado por al menos una instalación y 1.8 es la restricción de las variables de decisión.

1.3.3. Programación Lineal Entera (*Linear Integer Programming* - LIP)

En Love y Wong (1976), se propone formular el problema de *FLP* mediante un programa entero mixto (*Mixed Integer Programming* por sus siglas en inglés *MIP*) del *QAP*.

Se hacen las siguientes consideraciones:

- Las localizaciones están dadas como puntos de dos dimensiones en el plano.
- Los costos de transporte son proporcionales a las longitudes.

Definición de variables:

$$h_{ik}^r = \begin{cases} 1 & \text{si existe una distancia horizontal entre las instalaciones } i \text{ y } k \text{ cuando} \\ & \text{la instalación } i \text{ está a la derecha de la instalación } k \\ 0 & \text{cualquier otro caso} \end{cases}$$

$$h_{ik}^l = \begin{cases} 1 & \text{si existe una distancia horizontal entre las instalaciones } i \text{ y } k \text{ cuando} \\ & \text{la instalación } i \text{ está a la izquierda de la instalación } k \\ 0 & \text{cualquier otro caso} \end{cases}$$

$$v_{ik}^a = \begin{cases} 1 & \text{si existe una distancia vertical entre las instalaciones } i \text{ y } k \text{ cuando} \\ & \text{la instalación } i \text{ está por encima de la instalación } k \\ 0 & \text{cualquier otro caso} \end{cases}$$

$$v_{ik}^b = \begin{cases} 1 & \text{si existe una distancia vertical entre las instalaciones } i \text{ y } k \text{ cuando} \\ & \text{la instalación } i \text{ está por debajo de la instalación } k \\ 0 & \text{cualquier otro caso} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{si la instalación } i \text{ es asignada a la localidad } j \\ 0 & \text{cualquier otro caso} \end{cases}$$

$(\overline{x_i}, \overline{y_i})$: localización de la instalación i.

En la Figura 1.1 se observa que $h_{ik}^l = 0$, ya que el departamento i no se encuentra a la izquierda de la instalación k, por lo que no existe tal distancia, de igual forma para la distancia vertical v_{ik}^b, ésta tampoco existe porque el departamento i está ubicado por encima de la instalación k. Lo anterior se ve reflejado en la formulación, puesto que sólo puede existir una sóla distancia horizontal h_{ik} y una vertical v_{ik} . En la Figura 1.1 también se observa que $h_{ik}^r = \bar{x_i} - \bar{x_k}$ y $v_{ik}^a = \bar{y_i} - \bar{y_k}$, lo cual también considerado en la formulación.

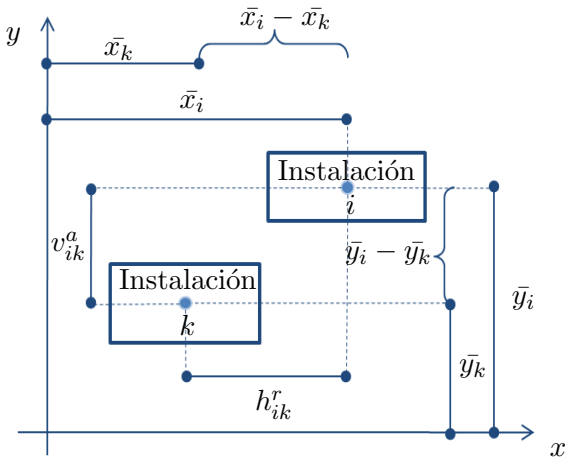


Figura 1.1: Formulación de (Love y Wong, 1976)
Fuente: Elaboración propia

Formulación:

$$Minimizar \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} - \sum_{i=1}^{n-1}\sum_{k=i+1}^{n} f_{ik}\left(h^r_{ik} + h^l_{ik} + v^a_{ik} + v^b_{ik}\right)$$

sujeto a:

$$h^r_{ik} - h^l_{ik} = \overline{x_i} - \overline{x_k}; \qquad i = 1, 2, 3, ..., n; \qquad k = i + 1$$

$$v^a_{ik} - v^b_{ik} = \overline{y_i} - \overline{y_k}; \qquad i = 1, 2, 3, ..., n; \qquad k = i + 1$$

$$\overline{x_i} + \overline{y_i} = \sum_{j=1}^{n} \left(\overline{x_j} + \overline{y_j}\right) x_{ij}; \qquad i = 1, 2, 3, ..., n$$

$$\overline{x_i} - \overline{y_i} = \sum_{j=1}^{n} \left(\overline{x_j} - \overline{y_j}\right) x_{ij}; \qquad i = 1, 2, 3, ..., n$$

$$\sum_{j=1}^{n} x_{ij} = 1; \qquad i = 1, 2, 3, ..., n$$

$$\sum_{i=1}^{n} x_{ij} = 1; \qquad j = 1, 2, 3, ..., n$$

$$x_{ij} \in [0,1]\,; \qquad i = 1,2,3,...,n$$

$$h_{ik}^{r} + h_{ik}^{l} + v_{ik}^{a} + v_{ik}^{b} \geq 0; \qquad i = 1,2,3,...,n; \qquad k = i+1$$

$$\overline{x_i}, \overline{y_i} \geq 0; \qquad i = 1,2,3,...,n$$

La formulación anterior tiene n^2 variables enteras y $n^2 + 3n$ restricciones, se hace complicado modelar problemas con nueve o más instalaciones.

En general, entre 1960 - 1980 la atención fue dirigida a la formulación, mediante diversas representaciones y aproximaciones teóricas. A partir de la década de los 90's se puso más atención a la solución del problema de optimización (Sherali *et al.*, 2003); muchos investigadores han desarrollado técnicas heurísticas para dar solución al problema.

1.3.4. FLP es *NP-completo*

Como se vió anteriormente el *FLP* puede formularse como un *QAP*, el cual se ha demostrado pertenece a la clase NP-completo (Sahni y González, 1976) y es del tipo combinatorio, lo cual implica que, para instancias grandes, no puede ser resuelto por algún algoritmo en tiempo polinomial.

Naturaleza Combinatoria del problema (*FLP*)

- Se da un conjunto finito de objetos (instalaciones)
- Cada objeto puede tomar un rango de atributos (diferentes localizaciones)
- Se desarrolla una solución al problema fijando los valores de atributos para todos los objetos
- Sólo se permiten ciertas combinaciones de los valores de atributos, es decir, aunque existen diferentes conjuntos de localizaciones que serán solución, algunos de ellos no están permitidos dadas las restricciones de espacio, intercambio de materiales y traslape.

Tanto el problema de tipo unidimensional y bidimensional son díficiles de resolver de forma óptima. Se sabe que el modelo para ambos pertenecen a la clase *NP-completo* (Heragu, 2006).

Además el *FLP* puede transformarse polinomialmente al problema del agente viajero (Kusiak y S.S., 1987), por lo tanto el *FLP* es al menos tan difícil como el *TSP*, el cual es bien sabido que es NP-completo.

Todo problema *NP-completo* puede ser resuelto mediante una búsqueda exhaustiva. Desafortunadamente cuando el tamaño de la entrada crece (número de departamentos a instalar) el tiempo para realizar dicha búsqueda se vuelve rápidamente exponencial, incluso para instancias de tamaño relativamente pequeño.

Para varios problemas es posible diseñar algorimos exactos que son significativamente más rápidos que una búsqueda exhaustiva, aunque siguen siendo de tiempo no polinomial, como es el caso del algoritmo aquí propuesto.

La siguiente Figura 1.2 muestra las posibles combinaciones para una distribución de dos departamentos. Se tienen 16 posibilidades diferentes. La distancia entre los centros de cada instalación es la variable más importante, ya que de ésto dependerá la minimización de los costos. Las líneas marcadas con rojo, son distancias no óptimas, mientras que las líneas negras muestran una distribución en donde la distancia es la menor.

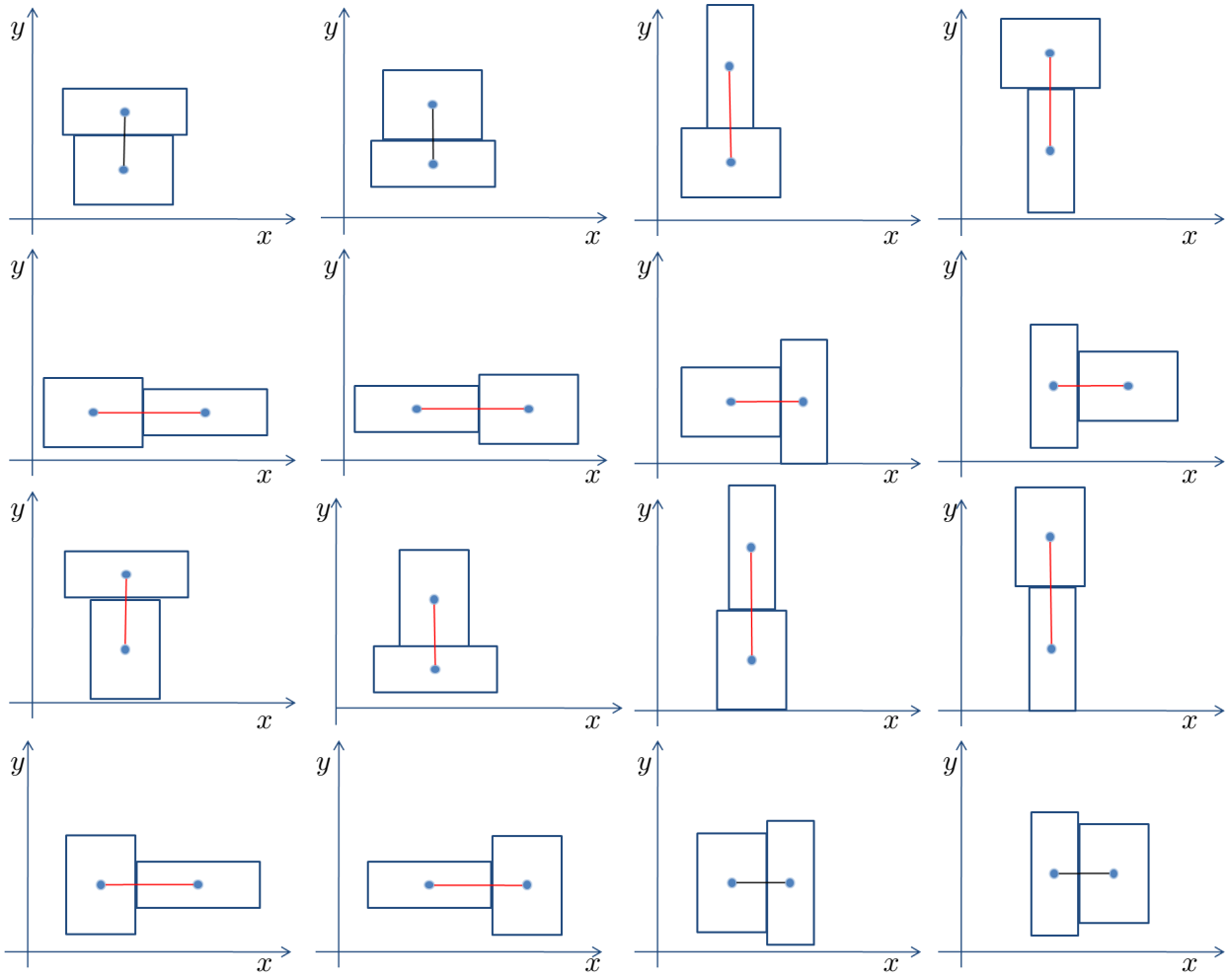


Figura 1.2: Arreglos posibles para una distribución que involucra dos instalaciones
Fuente: Elaboración propia

Ya que se ha definido el problema de Distribución de Planta a partir del enfoque de la programación matemática, a continuación se presentan los métodos de solución exactos para dicho problema, así como una introducción al algoritmo que se propone en este trabajo de investigación.

Capítulo 2

Métodos de Solución para el FLP

El problema de Distribución de Planta ha sido formalmente estudiado desde finales de 1940. Los primeros esfuerzos para resolverlo involucraron el uso de diagramas de flujo, esquemas de procesos y la experiencia y conocimiento de los analistas de la distribución.

Actualmente existen en el mercado paquetes para modelar, diseñar y analizar los problemas de distribución. Cada software tiene sus propias ventajas y desventajas, muchos de éstos paquetes recurren a las herramientas de simulación o no usan técnicas de solución sofisticadas como lo son los algoritmos exactos de optimización. Los programas más populares son de plataforma CAD (*computer-aided design*), por mencionar algunos: FactoryCAD, FactoryPLAN y FactoryFLOW.

Desde el punto de vista de la Investigación de Operaciones y la Optimización Matemática, se ha usado un algoritmo exacto para resolver las formulaciones matemáticas del problema de distribución presentadas anteriormente, éste es Ramificación y Acotamiento (Branch and Bound - $B\,\&\,B$). Aunque las aportaciones más significativas se han hecho utilizando algoritmos heurísticos y metaheurísticos Scholz *et al.* (2009) y Mir y H. (2001).

2.1. Relajación Lineal (RL)

Se llama así al problema lineal(Linear Program- LP) obtenido a partir del problema entero (Integer Program - IP), el cual se pretende resolver, mediante la omisión de la restricción de integralidad de las variables. El valor óptimo encontrado en la relajación lineal z_{LP} , en el caso de minimización, será una cota inferior del valor óptimo del problema entero z_{IP} , es decir, $z_{LP} \leq z_{IP}$. Si el número de restricciones del problema entero lineal es suficientemente pequeño, es posible utilizar un método clásico de solución, como lo es el algoritmo $B\,\&\,B$.

2.2. Branch and Bound *(B & B)*

Existen muchos problemas de optimización para los cuales no se conocen métodos de solución *directos*, o bien, son ineficientes. Los problemas pueden ser *difíciles* porque su función objetivo o las restricciones son no convexas, como lo es el caso del *FLP*, o porque algunas de las variables están restringidas a valores discretos (Lawler y D., 1966).

El algoritmo permite resolver problemas *dificiles* mediante la aplicación de métodos de solución para problemas *fáciles*, además ha esultado ser una de las mejores herramientas para la solución de problemas de optimización.

La idea general es dividir el problema en partes manejables. Si las soluciones de los casos más pequeños se obtienen de forma sencilla, entonces la solución del problema original puede ser obtenida por medio de una combinación de dichas soluciones. Si los casos más pequeños continuan siendo muy difíciles de resolver, entonces pueden ser divididos en instancias aún más pequeñas. Este proceso *recursivo* continua hasta que los problemas derivados son tan pequeños que fácilmente se puede obtener una solución.

Una forma típica de representar esta técnica es por medio de un árbol de enumeración. Un ejemplo de éste se muestra en la Figura 2.2.

La solución de un problema se obtiene yendo hacia abajo en el árbol y con las soluciones de los casos más pequeños.

Se debe diseñar una estrategía para cada problema a tratar, la cual debe de estar constituida de las siguientes partes:

- Dividir/Ramificar un problema en uno o más casos pequeños. Construir un conjunto finito y contable que contenga todas las soluciones del problema. Particionar recursivamente un conjunto no vacío de soluciones, en subconjuntos disjuntos.
- Conquistar/Resolver cada uno de los casos más pequeños. A menos que uno de los casos no sea lo suficientemente pequeño usar la recursión para realizar esto.
- Acotar. Si es necesario, combinar las soluciones de los casos más pequeños para obtener la solución del problema original. De lo contrario, eliminar las que no cumplen con los objetivos propuestos.

Para escoger la variable sobre la que se ramificará se tienen tres posibilidades "Regla de Ramificación"(Mora, 2009).

- La primera variable no entera encontrada, es decir, la variable no entera de subíndice menor.
- La variable no entera de mayor parte fraccionaria.

- La variable no entera "menos entera", es decir la variable más alejada de los dos enteros que la rodean, o lo que es lo mismo, la variable cuya parte fraccionaria está más cerca de 0.5.

Escoger el nodo sobre el que se realizará la búsqueda, escoger la rama del árbol.

- Búsqueda en Profundidad (*Depth-first search*). Se selecciona un hijo del nodo precedente después de ramificar y retrocede (moverse hacia atrás en el árbol de búsqueda para seleccionar el siguiente nodo) algunos nodos como sea posible después de podar el nodo.
- Búsqueda a lo Ancho (*Breadth-first search*). Se selecciona el nodo que está más cerca de la parte superior del árbol.
- Mejor Cota (*Best Bound search*). Se selecciona el nodo con la mejor cota inferior.

  Combinaciones de las anteriores:

  - Búsqueda a lo ancho para cierto número de nodos seguido de búsqueda en profundidad.
  - Búsqueda en profundidad mientras que se realice la ramificación y entonces utilizar la Mejor Cota después de podar el nodo.

Con frecuencia la elección de la regla se realiza de forma heurística. La Búsqueda en Profundidad permite obtener soluciones rápidas (existen más restricciones de las ramas en el árbol, y dichos límites ayudar a obtener las soluciones enteras), pero con el riesgo de que la calidad de la solución se vea afectada si se realizan malas ramificaciones. Por otro lado, la Búsqueda a lo Ancho rechaza el riesgo de trabajar en paralelo, por lo que en este trabajo se utiliza este tipo de búsqueda.

Después de estudiar un nodo, se pueden presentar tres casos:

- La solución no es entera, se escoge una variable para bifurcar y se crean dos nodos hijos.
- La solución es entera, se actualiza el valor de la función objetivo y se descarta el nodo.
- No hay solución y se sondea el nodo.

En 1960 Land y Doing desarrollaron un mejor enfoque de este algoritmo para los problemas *MIP*, posteriormente Gilmore (1962) y Lawler (1963) independientemente resolvieron el *QAP* (problema base para el *FLP*). El algoritmo inicia con la solución de la relajación lineal, si la solución encontrada es entera el algoritmo termina. En otro caso se escoge una variable x_i con valor fraccional x_i^* y se resuelven dos sub-problemas, uno en

donde al problema original se le añade la restricción $x_i \geq \lfloor x_i^* \rfloor$ (donde $\lfloor x_i^* \rfloor$ es el menor entero mayor que x_i^*) y al segundo problema se le añade la restricción $x_i \geq \lceil x_i^* \rceil$ (donde $\lceil x_i^* \rceil$ es el entero más grande menor que x_i^*).

El algoritmo continúa construyendo nuevos problemas lineales siguiendo una estructura de árbol en donde cada nodo representa los subproblemas a resolver (*branching*). Una ramificación puede ser acotada (*bounding*) si es una solución óptima entera o si el valor de la solución óptima es mayor que la actual cota, esto es, mejora la solución actual entera.

Su principal ventaja es que elimina selectivamente la gran mayoría de soluciones incluso antes de buscarlas. La desventaja de esta técnica se da en los problemas en los que la relajación lineal no se aproxima al conjunto convexo más pequeño que contiene todos los puntos factibles enteros, por lo que se generan una gran cantidad de subproblemas y se tiene que explorar todos los posibles diseños hasta concluir que no existe solución.

Cuando el número de restricciones lineales es grande o cuando la relajación lineal es reforzada mediante la adición de desigualdades válidas, generalmente se tiene un problema de tamaño exponencial, por lo tanto el sistema de restricciones no puede abordarse fácilmente; se recurre entonces a la técnica de planos de corte para resolver el IP.

Algunas razones importantes para aceptar ampliamente el algoritmo *(B & B)* son las siguientes:

1. El método es conceptualmente simple y fácil de entender
2. Es fácilmente adaptable a un amplio rango de situaciones problemáticas
3. Es fácilmente adaptable para su implantación computacional

A continuación se muestra un ejemplo del funcionamiento del algoritmo.

$$
\begin{aligned}
Maximizar \quad z \quad &= \quad 5x_1 + x_2 \\
sujeto \ a \quad &: \\
& \quad -x_1 + 2x_2 \leq 4 \qquad (2.1) \\
& \quad x_1 - x_2 \leq 1 \qquad (2.2) \\
& \quad 4x_1 + x_2 \leq 12 \\
& \quad x_1, x_2 \geq 0 \text{ y entero}
\end{aligned}
$$

En un problema de maximización, se inicia con un valor de la función objetivo igual a $-\infty$, porque cualquier solución de la RL cumplirá con el objetivo de maximizar.

Se resuelve la relajación lineal, la cual corresponde la nodo 0 del árbol de busqueda y cuya solución es $z = 14.6$ y $x_1 = 2.6$, $x_2 = 1.6$. Para este ejemplo, arbitrariamente se escoje la variable de decisión para ramificar, en este caso se eligió x_1, ver Figura 2.1

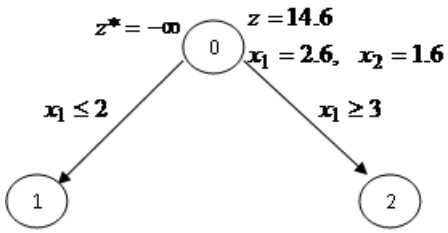


Figura 2.1: Árbol de Búsqueda del algorimo B & B

La solución para estos nuevos subproblemas se muestra a continuación,

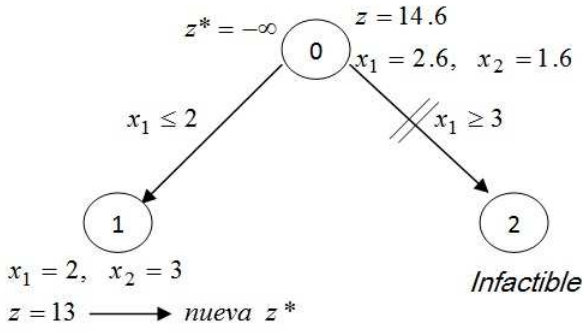


Figura 2.2: Árbol de Búsqueda del algorimo B & B

Dado que el nuevo valor de la función objetivo es 13 y $13 > -\infty$, entonces la solución al problema es $z = 13$ y $x_1 = 2$, $x_2 = 3$

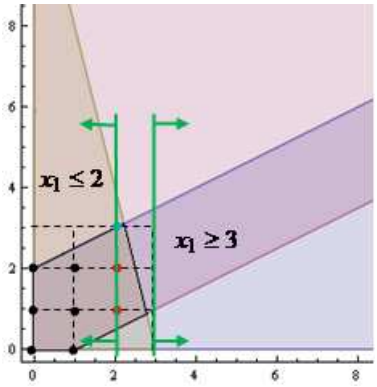


Figura 2.3: Gráfica del problema. Representación de las restricciones agregadas al problema original

2.3. Planos de Corte

Varios problemas de optimización combinatoria tienen formulaciones naturales como problemas lineales. La region factible de tales formulaciones puede tener puntos extremos con componentes fraccionales, en tal caso, la solución optima no podrá ser obtenida por medio de la solución de la Relajación Lineal. Sin embargo, si tenemos una descripción concisa de la cubierta convexa de los puntos factibles, por medio de desigualdades lineales, podemos resolver tal problema lineal entero utilizando un problema lineal. Encontrar dicha descripción, es lo que ahora nos interesa.

Un *plano de corte*, es una desigualdad lineal que es generada como se vaya necesitando en el curso de ir resolviendo un problema lineal entero como una secuancia de problemas lineales.

Conforme vayamos agregando desigualdades a nuestra Relajación Lineal, la secuencia de los valores optimos de los problemas lineales, es una secuencia no creciente de límites superiores del valor óptimo para el problema entero. La dificultad de aplicar el método de planos de corte reside en la problemática de encontrar las desigualdades.

Método General de Planos de Corte.

Paso 0. Iniciar, permitir RL ser la relajación lineal del problema entero IP

Paso 1. Hacer que x^* sea el punto extremo solución de RL

Paso 2. Si x^* es entero, entonces terminar, porque x^* es el óptimo de IP.

Paso 3. Si x^* no es entero, entonces encontrar una desigualdad que cumpla con todas las soluciones factibles posibles del IP, pero que sea violada por x^*, agregar la desigualdad a la RL y regresar al Paso 1.

A finales de la década de los 50's Gomory desarrolló el método de planos de corte (Graves y Wolfe, 1953), proponiendo una forma sencilla de derivar cortes de un problema de programación entra mediante la utilización de información asociada con sus restricciones lineales. Experiencia práctica con el algoritmo de Gomory muestra que la calidad de los cortes generados es pobre conforme avanza. En una larga secuencia de iteraciones no se obtiene una mejoría significativa.

La idea de este enfoque consiste en ir resolviendo una sucesión LP relajados con una región factible cada vez más restringida, hasta alcanzar una solución óptima con valores enteros, entonces, si la solución óptima del LP relajado no tiene todas sus componentes enteras, es posible obtener una desigualdad (corte o restricción) válida para la región factible del IP. Ésta se genera partiendo del principio de que todo número real k puede expresarse como la suma de su parte entera $\lfloor k \rfloor$ y su parte fraccional f_i , $k = \lfloor k \rfloor + f_i$,

entonces de la tabla óptima del Método Simplex, se elige una restricción del problema y se expresa como la suma de variables de sus partes enteras y sus partes decimales:

$$\lfloor x_i^* \rfloor + f_i = x_i + \sum_{j=m+1}^{n} \left(\lfloor a_{ij} \rfloor + f_{ij} \right) x_j = x_i + \sum_{j=m+1}^{n} \lfloor a_{ij} \rfloor \, x_j + \sum_{j=m+1}^{n} f_{ij} x_j$$

donde, a_{ij}: es el coeficiente asociado con la variable de decisión x_j y $a_{ij} = \lfloor a_{aij} + f_{ij} \rfloor$.

x_i^*: componentes de la matriz x^* de las variables de decisión, $x_i^* = \lfloor x_i^* \rfloor + f_i$.

Se agrupan las partes enteras y las partes fraccionales y dado que f_i , $f_{ij} \in (0,1)$ y $\sum_{j=m+1}^{n} f_{ij} x_j \geq 0$ son las partes fraccionales de las variables se forma la nueva restricción:

$$-\sum_{j=m+1}^{n} f_{ij} x_j \leq -f_i$$

Entonces, se añade ésta restricción a la tabla del problema, con su respectiva variable de holgura, y se resuelve mediante la utilización del método dual simplex, al procedimiento anterior se le conoce como "algoritmo de separación". Planos de corte presenta algunas desventajas como: convergencia lenta a la solución debido al incremento de variables, además de que supone cierta dificultad en escoger qué restricción separar; el método dual no genera soluciones factibles al problema, sino hasta que llega a la solución.

∗ Ejemplo **1** *Utilizando el algoritmo fraccional de Gomory para resolver el siguiente problema entero se tiene que,*

$$Maximizar \quad 7x_1 + 6x_2$$

sujeto a:

$$\begin{aligned} 2x_1 + 3x_2 &\leq 12 \\ 6x_1 + 5x_2 &\leq 30 \\ x_{1,}x_2 &\geq 0 \\ x_{1,}x_2 &\in \mathbb{Z} \end{aligned}$$

Paso 1. Resolviendo la relajación lineal del problema, se obtiene que $x_1 = 3.75$ y $x_2 = 1.5$ que no son enteros, por lo que se continúa con el Paso 3. Se muestra la tabla óptima de la relajación lineal.

Base	x_1	x_2	s_1	s_2	RHS
z	0	0	0.125	1.125	32.25
x_2	0	1	0.75	-0.25	1.5
x_1	1	0	-0.625	0.375	3.75

Tabla 2.1: Tabla óptima 1 del ejemplo 1.

Generamos el corte con el renglón de la variable x_1, la que tiene mayor $(x_{B_i} - \lfloor x_{B_i} \rfloor)$, esto es,

$$-0.625s_1 + 0.375s_2 \geq 3.75 \Rightarrow (-1 + 0.375)s_1 + 0.375s_2 \geq 3.75 - 3$$

$$\Rightarrow 0.375s_1 + 0.375s_2 \geq 0.75$$

y en términos de las variables de decisión tenemos,

$$3x_1 + 3x_2 \leq 15$$

restricción que se agrega al problema lineal del ejemplo 1. Resolviendo nuevamente, se tiene que $x_1 = 5$ y $x_2 = 0$, lo cual satisface la restricción de integralidad del problema entero del ejemplo.

En este caso fue suficiente con generar un corte para llegar a la solución óptima, mientras que utilizando $B \,\&\, B$ se realizan 7 iteraciones.

2.4. Estado del Arte

Varios autores han realizado diversas formulaciones matemáticas del problema de distribución de planta, o bien, variaciones de las antes vistas, que obedecen a características específicas del problema a resolver, así como diversas técnicas de solución.

En Heragu y A. (1992) se utilizó el algoritmo de Recocido Simulado (*RS*) y un algoritmo híbrido en problemas unidimensionales con áreas desiguales y bidimensionales con áreas iguales. Para el primer caso, el algoritmo reportó mejores soluciones que las encontradas anteriormente, para todos los ejemplos utilizados. En el caso bidimensional con áreas iguales, el 80 % de los ejemplos mejoró la solución antes reportada en la literatura.

En Meller *et al.* (1998) se reformuló el modelo de Montreuil mediante una redefinición de las variables binarias y ajustando el área de los departamentos mediante la inserción de restricciones válidas. Utilizan el algoritmo B & B y son capaces de incrementar el número de problemas solucionables, pero aún insuficiente para que sea de interés práctico.

Mir y H. (2001) trabajaron con el problema de áreas desiguales bidimensional involucrando 50-100 departamentos, aunque el costo computacional es alto, es el primer trabajo que puede modelar dicho número de instancias además de obtener soluciones de calidad. Implementaron el algoritmo de recocido simulado *(RS)* y algoritmos genéticos; una característica especial de dicho trabajo es la división que se hace de cada instalación en *células unitarias* con el fin de situar instalaciones de igual tamaño.

Se desarrolló un algoritmo genético para áreas desiguales en Honiden (2004). A modo de prueba utilizaron dos diferentes tamaños, 10 y 20 instalaciones, el algoritmo se corrió 5 veces y cada corrida tomó un tiempo de 3.5 minutos con 500 generaciones; consideraron que las soluciones obtenidas eran buenas, ya que la mayoría cayó dentro de cierto espacio de solución.

Amaral (2008) consideró el arreglo en línea (unidimensional), propuso una formulación *MIP* y presentó un algoritmo exacto B & B para su solución. Obtuvo una solución para 15 instancias en un tiempo de 39 minutos aproximadamente, comparó sus resultados con los obtenidos con la formulación de Love y Wong (1976) en la cual se invirtieron 382 minutos para encontrar una solución.

En Tavakkoli-Moghaddam *et al.* (2007), se considera la demanda estocástica entre departamentos. Se utilizó el paquete Lingo con el algoritmo B & B para verificar la validez del modelo.

En Solimanpur y Jafari (2008) desarrollaron el algoritmo B & B para resolver el problema bidimensional de distribución de planta. Modelaron un problema pequeño, 4 instancias; un problema mediano, 15 instancias y un problema grande, 30 instancias; el algoritmo resultó eficiente para el problema pequeño y mediano, pero pobre para el tipo grande.

La utilización de un algoritmo de Búsqueda Tabú y Slicing Tree (cada nodo del árbol representa una instalación) se muestra en Scholz *et al.* (2009). Se pudo modelar un problema de 62 instancias bidimensional con áreas diferentes; se realizaron 10 corridas del algoritmo y se obtuvo una mejoría del 1.72 % comparando con otras formulaciones.

La técnica metaheurística más reciente para resolver el *FLP* es el algoritmo de colonia de hormigas (Wong, 2010). El autor afirma que dicho algoritmo es más robusto y tiene un mejor desempeño que Scholz *et al.* (2009). Se consideran departamentos de áreas desiguales y se logra utilizar hasta 62 instalaciones.

AUTOR	ARREGLO EN		ÁREAS		ALGORITMO	INSTALACIONES
	R	R^2	$=$	$\neq$		
Mir y H. (2001)		✓		✓	RS y A. Genético	50 y 100
Honiden (2004)		✓		✓	A. Genético	10 y 20
Amaral (2008)	✓			✓	B & B	15
Solimanpur y Jafari (2008)		✓		✓	B & B	4 y 15
Scholz *et al.* (2009)		✓		✓	B. Tabú y S. Tree	62
Wong (2010)		✓		✓	Colonia Hormigas	62

Tabla 2.2: Resumen del Estado del Arte para el *FLP*.

Como se mencionó anteriormente, desde su aparición, el algoritmo B & C se volvió rapidamente una de las técnicas más populares para la solución de problemas de Programación Entera Mixta.

El algoritmo B & C ha sido el responsable de resolver problemas de optimización combinatoria muy importantes y en particular los que se mencionan a continuación.

En Padberg y G. (1991) se prueba que el algoritmo puede ser utilizado para resolver problemas de gran tamaño de forma óptima. Los autores aseguran que conforme los paquetes para resolver la relajación lineal vayan mejorando, ya que la mayoría del tiempo es invertido en esta parte, se tendrán avances en la generación de restricciones y se mejorará el tiempo total de corrida del algoritmo. La siguiente tabla muestra algunos resultados importantes obtenidos para el problema del agente viajero para diferentes instancias. La primera columna indica el número de ciudades involucradas en el problema, la segunda columna muestra la cantidad de cortes necesarios en todo el árbol de búsqueda B & B y finalmente, la última columna presenta el número de nodos explorados por el algoritmo B & C para encontrar la solución óptima.

El problema de Programación de Camiones Despachadores, en inglés *Truck Dispatching Scheduling problem - TDS*, es un problema de ruteo y programación de tareas que fue resuelto mediante el algoritmo B & C de forma eficiente (Bixby y Lee, 1998), ya que otras

CIUDADES	**CORTES**	***B* & *C* NODOS**
532	4887	45
1002	4988	20
2392	6963	53

Tabla 2.3: Resultados del *TSP* con *B* & *C*
Fuente:Padberg y G. (1991).

técnicas sólo podían encontrar buenas soluciones factibles. Los autores decidieron utilizar *B* & *C* a diferencia de *B* & *B*, el cual también proporciona el óptimo, debido a que la experiencia señala que el desempeño del *B* & *C* es mejor para instancias grandes de los problemas.

En Bixby y Lee (1998) se muestra como *B* & *C* resuelve el problema de forma óptima en una cantidad de tiempo razonable. Se prueban varios tipos de problemas en donde algunos de los resultados fueron:

PROBLEMA	*B* & *B*nodos	*B* & *C* nodos	*B* & *B* CPU time	*B* & *C* CPU time
TIPO 1	62,644	1	340,690.0	3,702. 5
TIPO 2	55,460	1	333,047.0	1,190.4
TIPO 3	43,990	198	156,158.0	5,624.3

Tabla 2.4: Comparación entre *B* & *B* y *B* & *C* para el *TDS*
Fuente:Bixby y Lee (1998).

Durante los últimos 50 años el problema de Programación de la Tripulación, en inglés *Crew Scheduling Problem*, en donde una serie de viajes se asigna a algunos equipos mecánicos disponibles, el objetivo es minimizar los costos de asignar un subconjunto de los viajes a cada equipo, de tal manera que ningún viaje quede sin asignar, mientras se satisfacen varias restricciones impuestas por el gobierno y leyes de trabajo. El problema ha sido continuamente estudiado, pero siempre se han dado soluciones aproximadas, incluso para flotillas de aviones pequeñas. De los 52 tipos diferentes de problemas probados, el más largo (85,552 variables) fue resuelto en 1.5 horas, tiempo que incluye la lectura del problema y la eliminación de columnas duplicadas, además sólo tres de los problemas requirieron ramificar y el árbol de búsqueda más largo fue de 4 nodos.

2.5. Introducción al algoritmo Branch and Cut (*B* & *C*).

Si con los métodos exactos anteriores no se logra obtener una solución óptima, se puede descomponer el IP en dos sub-problemas, por ejemplo, añadiendo cotas superiores e inferiores de las variables que actualmente tienen valores fraccionales (*branching*), entonces,

podemos resolver cada nodo de forma recursiva con el algoritmo de planos de corte, donde la enumeración del árbol es controlada por el algoritmo B & B. El método que combina éstas dos técnicas es llamado *Branch and Cut algorithm*, algoritmo de Ramificación y Corte ($B\,\&\,C$).

Dicho método fue introducido a finales de los años 80´s por Padberg y G. (1991), donde se abordó el *TSP* considerando 532 ciudades. Además, demostraron que este método es el más prometedor para obtener soluciones exactas.

El esquema general de solución es generar cortes globalmente válidos (tanto para los nodos en donde se introduce, como para el problema original) en cada nodo del árbol de ramificación para obtener una secuencia de aproximaciones a la cubierta convexa del programa entero, esto es, ajustarse cada vez mejor al conjunto de soluciones factibles.

Ambas técnicas se benefician una de la otra, por un lado la cota producida en cada nodo del árbol es mejor que en B & B debido a las nuevas desigualdades que son añadidas a la formulación del correspondiente sub-problema; aunque la solución de cada nodo requiere más esfuerzo, en diversas aplicaciones prácticas se ha constatado que tal esfuerzo deriva en un menor número de nodos que necesitan ser explorados permitiendo, la resolución de problemas difíciles que las técnicas anteriores aisladamente no resuelven (Dell´Amico *et al.*, 1997). Con respecto a planos de corte, se recurre a una nueva ramificación tan pronto como se detecte una convergencia lenta, cuando no se obtenga un incremento significativo (por ejemplo de 0.01 %) en el valor óptimo del LP; entonces, se genera un nuevo subproblema en lugar de generar más cortes, este fenómeno es llamado *"tailing-off"*. Por lo tanto, la convergencia total se asegura ramificando, pero la efectividad de esta técnica depende fuertemente de la calidad de los cortes generados.

Una de las claves principales para el buen diseño de un algoritmo $B\,\&\,C$, consiste en saber cómo ir incorporando las distintas desigualdades lineales en la relajación de cada subproblema, sin que exista la necesidad de incorporar cada una de las restricciones. Así que, hacer funcionar eficientemente todo el mecanismo no es sencillo, dado que constantemente se añaden más y más restricciones. Conviene eliminar periódicamente aquellas que han dejado de ser útiles en las últimas iteraciones, aunque no siempre conviene olvidar totalmente dichas restricciones porque pueden volver a ser útiles. Las desigualdades que no mejoran la aproximación de la cubierta convexa del problema pueden ser removidas. Para evitar remover *buenas* desigualdades se almacenan temporalmente en una estructura llamada *pool*. Luego, cada vez que se pretenda fortalecer la relajación lineal de un nodo, antes de llamar al algoritmo de separación, conviene examinar la estructura pool para ver si hay en ella algún corte, razón por la cual las restricciones deben ser globalmente válidas.

Las variables pueden ser controladas dinámicamente. Con frecuencia la optimización de problemas incluye un gran número de variables, entonces el número de variables iguales a cero en la solución factible es escaso. Por ejemplo, en el *TSP* con n ciudades, una solución factible tiene variables diferentes de cero y el número total de variables es $n(n-1)/2$.

Es posible que el algoritmo comience con un conjunto de variables probablemente más relevantes y las nuevas variables son incorporadas sólo si es necesario.

La técnica de $B \,\&\, C$ conlleva una gestión inteligente de modelos matemáticos con una buena administración de restricciones.

El algoritmo permite manejar grandes cantidades de variables y restricciones por lo que es usado en bastantes y diversas aplicaciones obteniendo buenos resultados, por lo que se sugiere sea utilizado en el problema de distribución de planta (Dell´Amico *et al.*, 1997).

En este trabajo de investigación, se propone la utilización del algoritmo $B \,\&\, C$ como metodología de solución. Se diseñará un algoritmo para un problema de distribución de planta con características definidas.

Hasta este momento se tiene un panorama general del algoritmo y sus resultados en otras aplicaciones y se definen los métods de sólición que conforman el algoritmo $B \,\&\, C$. En el siguiente capítulo se presenta el diseño del algoritmo propuesto para resolver el problema de Distribución de Planta.

Capítulo 3

Diseño del Algoritmo propuesto

En este capítulo se define el modelo y características del problema *FLP* a resolver, posteriormente se presenta el algoritmo *B & C* propuesto y cada uno de sus elementos, además se presentan algunas de las adaptaciones necesarias para su mejor funcionamiento.

3.1. Definición del modelo de Distribución de Planta

El problema bidimensional de áreas desiguales, se define como aquella distribución de planta que se realizará considerando la disposición de departamentos a lo largo del plano bidimensional. La información acerca de los productos (bienes o servicios) y procesos (productivos o de información) son ya conocidos al momento del diseño de la distribución de planta. Además para el caso de procesos productivos, se considera un volumen y variedad de productos constante.

Se sabe de antemano, que no todos los departamentos a localizar tendrán la misma área, por lo que, para fines prácticos, su superficie se aproximará a la forma geométrica más cercana, en este caso, cuadrangular o rectangular. Por ejemplo, si consideramos la Figura 3.1, se observa que podemos manejar los departamentos como áreas de forma rectangular, dicha aproximación nos proporcionará un manejo más fácil de instalaciones y la calidad de nuestra solución no se verá drásticamente afectada.

Esta suposición nos ayuda a poder formular el problema de distribución de planta mediante el modelo ABSMODEL 3 propuesto por (Kusiak y S.S., 1987) el cual se describe a continuación.

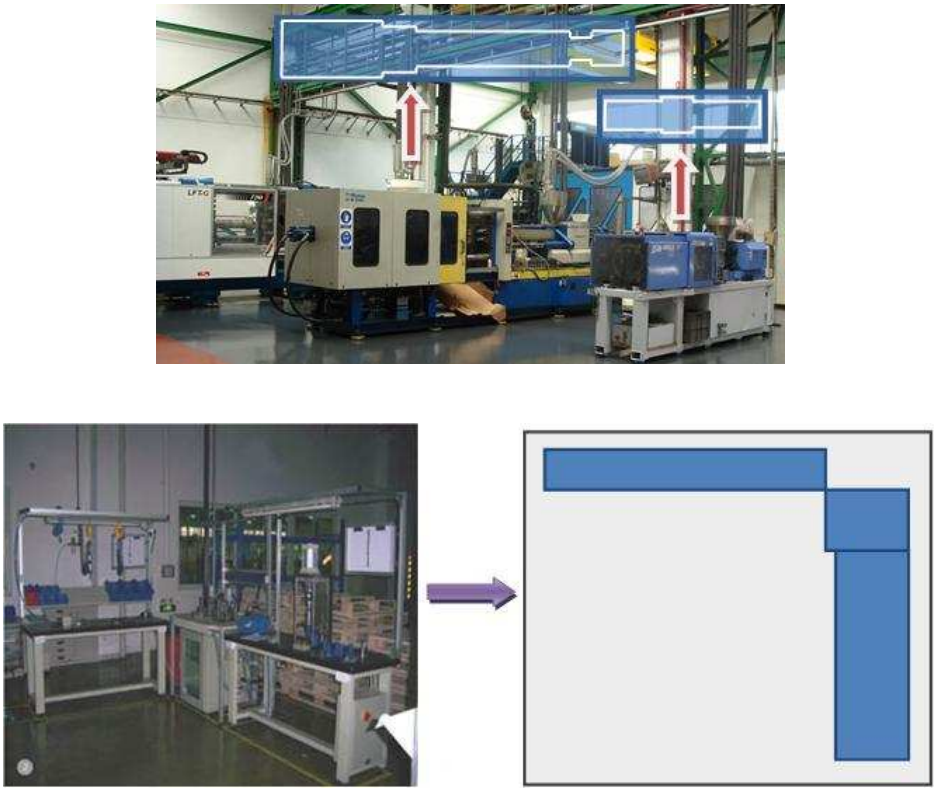


Figura 3.1: Aproximación de las áreas de las estaciones de trabajo.
Fuente: Elaboración propia

Parámetros y variables que involucra el modelo.

n : número de departamentos en el problema.

c_{ij} : costo de transportar una unidad de material por unidad de distancia entre los departamentos i y j.

f_{ij} : número de unidades a transportar entre los departamentos i y j.

l_i : largo del lado horizontal del departamento i (m).

b_i : largo del lado vertical del departamento i (m).

dh_{ij} : distancia mínima por la cual los departamento i y j pueden estar separados horizontalmente (m).

dv_{ij} : distancia mínima por la cual los departamento i y j pueden estar separados verticalmente (m).

H : dimensión horizontal de la planta.

V : dimensión vertical de la planta.

x_i : distancia horizontal entre el centro del departamento i y la línea vertical de referencia (VRL).

y_i : distancia vertical entre el centro del departmaneto i y la línea horizontal de referencia (VRL).

La siguiente figura muestra los parámetros y variables del modelo.

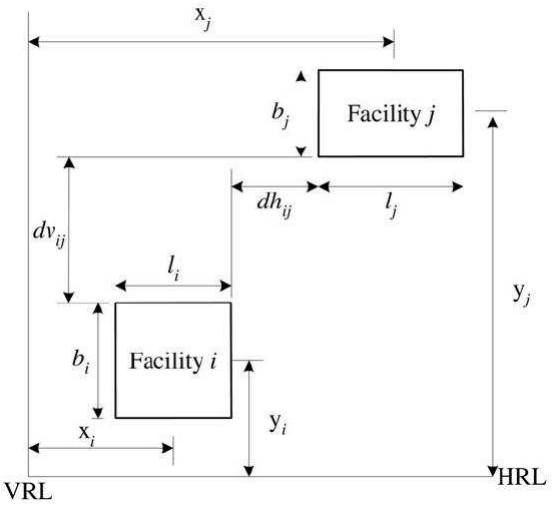


Figura 3.2: Diagrama de las variables de decisión y parámetros para un FLP con áreas desiguales bidimensional. Fuente: Heragu (2006)

El modelo queda como:

$$Minimizar \quad \sum_{i=1}^{n-1} \sum_{j=1+1}^{n} c_{ij} f_{ij} \left(|x_i - x_j| + |y_i - y_j| \right) \tag{3.1}$$

sujeto a:

$$|x_i - x_j| + M z_{ij} \geq 0.5 \left(l_i + l_j \right) + dh_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n \tag{3.2}$$

$$|y_i - y_j| + M \left(1 - z_{ij} \right) \geq 0.5 \left(b_i + b_j \right) + dv_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n \tag{3.3}$$

$$z_{ij} \left(1 - z_{ij} \right) = 0; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n \tag{3.4}$$

$$H \geq x_i \geq 1, \qquad i = 1, 2, 3, ..., n \tag{3.5}$$

$$V \geq y_i \geq 1, \qquad i = 1, 2, 3, ..., n \tag{3.6}$$

$$x_i, y_j, z_{ij} \geq 0 \tag{3.7}$$

La expresión 3.1 minimiza el costo total generado en el transporte de material entre departamentos. La restricciones 3.2 y 3.3 garantizan que los departamentos no se traslapen horizontal y tampoco verticalmente. La restricción 3.4 asegura que la variable z_{ij} tome valores de 0 ó 1, es decir es una variable binaria, la cual a su vez permite que sólo una de las restricciones 3.2 y 3.3 se mantenga. Además, no es necesario incluir restricciones de integralidad para las variables x_i y y_i . Si las dimensiones del espacio disponible para disponer el arreglo de departamentos son conocidas, entonces se agregan las restricciones 3.5 y 3.6 , que hacen que la distribución sea tal que esté dentro de los límites físicos de la planta.

Cabe señalar que se supone que el transporte de material entre departamentos ocurre a lo largo de un conjunto de caminos paralelos a las instalaciones, es decir, se tienen pasillos ortogonales o bien perpendiculares. El movimiento de material se muestra en la Figura 3.3.

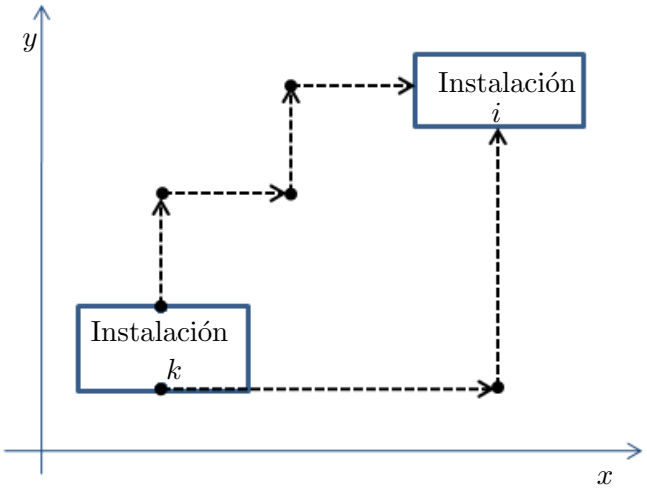


Figura 3.3: Movimiento en el transporte de materiales entre instalaciones.
Fuente: Elaboración propia

La diferencia fundamental entre el *QAP* y el ABSMODEL, es que el primero requiere del conocimiento previo de las ubicaciones en donde se localizarán los departamentos y se

trata de una formulación discreta, a diferencia del ABSMODEL que es una fomulación del tipo contínua. Esta diferencia hace poco atractivo al QAP y se considera más general al ABSMODEL.

Sabemos que la solución para un problema de programación lineal se encuentra en un punto extremo del conjunto convexo de soluciones posibles. Aunque para un problema de programación no lineal, la región factible sea un conjunto convexo, su solución óptima no tiene que ser un punto extremo de la región factible, más aún, en muchos casos, su región factible no será un conjunto convexo.

La condición de conjunto convexo no es satisfecha por el ABSMODEL debido a la función valor absoluto, esta formulación es un problema de programación no lineal. Por ejemplo, si tenemos dos departamentos con dimensiones horizontales 4 y 6 unidades respectivamente, la restricción 3.2 del modelo queda de la siguiente forma: $|x_1 - x_2| \geq 0.5(4+6)$ suponiendo que $dh_{12} = 0$, $z_{ij} = 0$ y la región factible en este caso será la mostrada en la Figura 3.4 y trazando una línea entre cualesquiera dos puntos, observamos que no se cumple con el concepto de conjunto convexo.

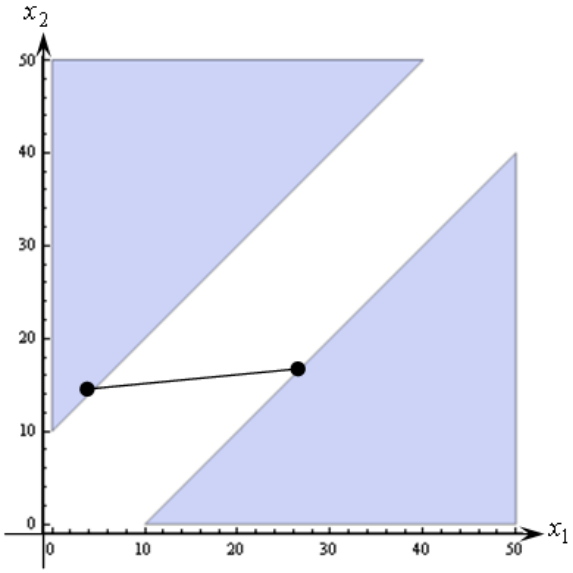


Figura 3.4: Prueba de la no convexidad de dos puntos factibles.
Fuente: Elaboración propia

Para linealizar el modelo anteriormente presentado se introducen las siguientes variables de decisión:

$$x_{ij}^{+} = \begin{cases} x_i - x_j & \text{si} \quad (x_i - x_j) > 0 \\ 0 & \text{cualquier otro caso} \end{cases} \tag{3.8}$$

$$x_{ij}^{-} = \begin{cases} x_i - x_j & \text{si} \quad (x_i - x_j) \leq 0 \\ 0 & \text{cualquier otro caso} \end{cases} \tag{3.9}$$

Por lo tanto $|x_i - x_j|$ puede ser sustituido por 3.8 y 3.9, esto es,

$$Minimizar \sum_{i=1}^{n-1} \sum_{j=1+1}^{n} c_{ij} f_{ij} \left(\left| x_{ij}^{+} + x_{ij}^{-} \right| + \left| y_{ij}^{+} + y_{ij}^{-} \right| \right) \tag{3.10}$$

sujeto a:

$$\left(x_{ij}^{+} + x_{ij}^{-} \right) + M z_{ij} \geq 0.5 \left(l_i + l_j \right) + d h_{ij}, \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n \tag{3.11}$$

$$\left(y_{ij}^{+} + y_{ij}^{-} \right) + M \left(1 - z_{ij} \right) \geq 0.5 \left(b_i + b_j \right) + d v_{ij}, \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n \tag{3.12}$$

$$x_i - x_j = x_{ij}^{+} - x_{ij}^{-} \tag{3.13}$$

$$y_i - y_j = y_{ij}^{+} - y_{ij}^{-} \tag{3.14}$$

Se mantienen las restricciones 3.4, 3.5, 3.6 y 3.7 de la formulación anterior.

En este modelo, uno de los valores ya sea x_{ij}^{+} ó x_{ij}^{-} debe de ser igual a cero, ya que esto posiciona al departamento i a la derecha o a la izquierda del departamento j de referencia; dichas variables representan el movimiento de la instalación hacia la parte positiva o negativa con respecto a otra instalación, por supuesto que una instalación no puede estar en ambas posiciones al mismo tiempo, la instalación no se puede dividir.

Para evitar que suceda lo anterior, que ambas variables tomen valores en la solución factible, y además asegurar el no traslape de departamentos, se modifica el modelo anterior, quedando de la siguiene forma:

$$Minimizar \sum_{i=1}^{n-1} \sum_{j=1+1}^{n} c_{ij} f_{ij} \left[x_{ij}^{+} + x_{ij}^{-} + y_{ij}^{+} + y_{ij}^{-} \right]$$

sujeto a:

$$x_i - x_j + Mp_{ij} + Mq_{ij} \geq 0.5\,(l_i + l_j) + dh_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$x_j - xi + Mp_{ij} + M\,(1 - q_{ij}) \geq 0.5\,(l_i + l_j) + dh_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$y_i - y_j + M\,(1 - p_{ij}) + Mq_{ij} \geq 0.5\,(b_i + b_j) + dv_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$y_j - y_i + M\,(1 - p_{ij}) + M\,(1 - q_{ij}) \geq 0.5\,(b_i + b_j) + dv_{ij}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$x_i - x_j = x_{ij}^{+} - x_{ij}^{-}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$y_i - y_j = y_{ij}^{+} - y_{ij}^{-}; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$x_i \leq H$$

$$y_i \leq V$$

$$p_{ij}, q_{ij} \in [0, 1]\,; \qquad i = 1, 2, 3, ..., n-1; \qquad j = i+1, ...n$$

$$x_i, y_i \geq 0, \qquad i = 1, 2, 3, ..., n$$

Se asume que el flujo de material se realiza entre los centros de cada departamento y el costo de mover una unidad de material es proporcional a la distancia. El modelo anterior se conoce en la literatura como LMIP 4 (*Linear Mixed Integer Programming 4*).

En general, el modelo ABSMODEL ha sido utilizado en Solimanpur y Jafari (2008) para resolver el *FLP* con ciertas características y adaptaciones, utilizando como algoritmo de solución *B & B*.

3.2. Diseño del Algoritmo B & C.

El problema de distribución de planta ha sido estudiado formalmente desde finales de los años 40's.

Desde la formulación del problema como el QAP se ha progresado en la solución del modelo. Un gran número de algoritmos han sido propuestos y se pueden dividir en:

- Algoritmos Exactos. Son de alto costo computacional, incrementan exponencialmente con respecto al tamaño de entrada por lo que ocupan gran memoria.
- Algoritmos Heurísticos. La ventaja de un algoritmo óptimo sobre un heurístico es que garantizan producir la mejor solución, o bien, una de las mejores soluciones cuando existe más de una, a diferencia de los heurísticos que no pueden asegurar el óptimo.

El algoritmo B & C se sitúa dentro de los algoritmos óptimos.

Antes de continuar, se explican detalladamente las dos partes fundamentales del algoritmo, Planos de corte y B & B.

En el caso de Planos de Corte se decidió utilizar el algoritmo que a continuación se presenta, porque éste considera que algunas variables de decisión deben ser continuas y otras enteras, a diferencia del algoritmo fraccional que no lo toma en cuenta.

3.2.1. Algoritmo Entero Mixto de Gomory.

Partiendo de una restricción en cualquier iteración,

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n + x_{n+i} = x_{B_i}$$

que puede reescribirse como,

$$\sum^{n} \left(a_{ij}^{+} + a_{ij}^{-}\right) x_j + x_{n+i} = (x_{B_i} - \lfloor x_{B_i} \rfloor) + \lfloor x_{B_i} \rfloor$$

donde,

$$a_{ij}^{+} = \begin{cases} a_{ij} & \text{si} \quad a_{ij} \geq 0 \\ 0 & \text{si} \quad a_{ij} < 0 \end{cases}, \quad a_{ij}^{-} = \begin{cases} 0 & \text{si} \quad a_{ij} \geq 0 \\ a_{ij} & \text{si} \quad a_{ij} < 0 \end{cases}$$

El algoritmo entero mixto procede de la misma forma que el algoritmo fraccionario, siendo la estructura del corte lo que difiere en ambos algoritmos.

El corte se genera de la siguiente forma,

$$\underbrace{\sum_{j=1}^{n} a_{ij}^{+} x_j + \sum_{j=1}^{n} \frac{h}{1-h}\left(a_{ij}^{-}\right) x_j}_{\text{para variables no enteras}} + \underbrace{\sum_{j=1}^{n} \left(a_{ij} - \lfloor a_{ij} \rfloor\right) x_j}_{\substack{\text{para variables enteras con} \\ (a_{ij} - \lfloor a_{ij} \rfloor) \leq h}} + \underbrace{\sum_{j=1}^{n} \frac{h}{1-h}\left(1 - a_{ij} - \lfloor a_{ij} \rfloor\right) x_j}_{\substack{\text{para variables enteras con} \\ (a_{ij} - \lfloor a_{ij} \rfloor) > h}} \geq h$$

donde $h = MAX(x_{B_i} - \lfloor x_{B_i} \rfloor)$.

Como se observa en la estructura de obtención del corte, dependiendo de la variable será la forma de obtener una restricción.

Considerando el siguiente problema entero:

∗ Ejemplo **2**

$$\begin{aligned} Maximizar \quad z &= 3x_1 - x_2 \\ sujeto \quad a &: \\ & 3x_1 - x_2 \leq 3 \\ & 5x_1 + 4x_2 \geq 10 \\ & 2x_1 + x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

Las variables de holgura quedan de la siguiente forma,

$$\begin{array}{rrrrrrrr} 3x_1 & -2x_2 & +s_1 & & & = & 3 \\ -5x_1 & -4x_2 & & +s_2 & & = & -10 \\ 2x_1 & +x_2 & & & +s_3 & = & 5 \end{array}$$

Resolviendo la relajación lineal se tiene la siguiente tabla óptima,

Base	x_1	x_2	s_1	s_2	s_3	RHS
z	0	0	0.7143	0	0.4286	4.2857
x_1	1	0	0.1429	0	0.2857	1.8571
x_2	0	1	-0.2857	0	0.4286	1.2857
s_2	0	0	-0.4286	1	3.1429	4.4289

Tabla 3.1: Tabla óptima 1 del ejemplo 2.

Considerando $h = MAX(x_{B_i} - \lfloor x_{B_i} \rfloor)$, generamos el primer corte con la restricción asociada a la variable x_1.
Esto es, $0.1429s_1 + 0.2857s_3 \geq 0.8571$ y en términos de las variables de decisición,

$$1.0001x_1 - 0.0001x_2 \leq 1.0001 \qquad \text{(Corte 1)}$$

Al agregar dicho corte y resolver nuevamente la relajación lineal, se tiene la siguiente tabla óptima.

Base	x_1	x_2	s_1	s_2	s_3	s_4	RHS
z	0	0	0	0.2499	0.4286	4.2490	1.7505
x_1	1	0	0	0	0	0.9998	1.0001
x_2	0	1	0	-0.25	0	-1.2497	1.2498
s_1	0	0	1	-0.4999	0	-5.4988	2.4993
s_3	0	0	0	0.25	1	-0.7498	1.7499

Tabla 3.2: Tabla óptima 2 del ejemplo 2 con el Corte 1.

Dado que aún no se tienen valores enteros para las variable de decisión, se genera un segundo corte con la restricción asociada a la función objetivo, ya que el
$MAX(x_{B_i} - \lfloor x_{B_i} \rfloor) = 1.7505 - 1 = 0.7505$.

$$0.2499s_2 + 4.2490s_4 \geq 0.7505$$

donde s_4 es la variable de holgura asociada al Corte 1, $s_4 = 1.0001 - 1.0001x_1 + 0.0001x_2$.

En términos de las variables de decisión,

$$2.99992x_1 - 1.00002x_2 \leq 0.99992 \qquad \text{(Corte 2)}$$

Agregando el plano de corte y resolviendo nuevamente, se tiene la siguiente tabla óptima.

Hasta este momento se tiene una solución que podría ser considerada como entera, generamos un corte más para ver que sucede. Esto se hace a partir de la restricción asociada a la variable de holgura s_3.

Base	x_1	x_2	s_1	s_2	s_3	s_4	s_5	RHS
z	0	0	0	0	0	0.0001	-1	1
x_1	1	0	0	0	0	1.0002	-0.0001	1.0002
x_2	0	1	0	0	0	3.0005	-1.0003	2.0006
s_1	0	0	1	0	0	3.0003	-2.0003	4.0005
s_2	0	0	0	1	0	17.0028	-4.0016	3.0032
s_3	0	0	0	0	1	-5.0009	1.0005	0.9990

Tabla 3.3: Tabla óptima 3 del ejemplo 2.

$$-5.0009s_4 + 1.0005s_5 \geq 0.9990 \rightarrow 4995.8991s_4 + 1.0005s_5 \geq 0.9990 \rightarrow$$

$$-4999.4x_1 + 1.50011x_2 \geq -4996.401 \qquad \text{(Corte 3)}$$

Se obtiene,

Base	x_1	x_2	s_1	s_2	s_3	s_4	s_5	s_6	RHS
z	0	0	0	0	0	0	-1	0	1
x_1	1	0	0	0	0	0	-0.0003	0.0002	1.0000
x_2	0	1	0	0	0	0	-1.0009	0.0006	2.0000
s_1	0	0	1	0	0	0	-2.0009	0.0006	3.9999
s_2	0	0	0	1	0	0	-4.0050	0.0034	2.9998
s_3	0	0	0	0	1	0	1.0015	-0.0010	1.0000
s_4	0	0	0	0	0	1	0.0002	-0.0002	0.0002

Tabla 3.4: Tabla óptima 4 del ejemplo 2.

Entonces, la solución óptima para el problema entero es $x_1 = 1$ y $x_2 = 2$. A continuación se muestra la representación gráfica de los cortes agregados al problema.

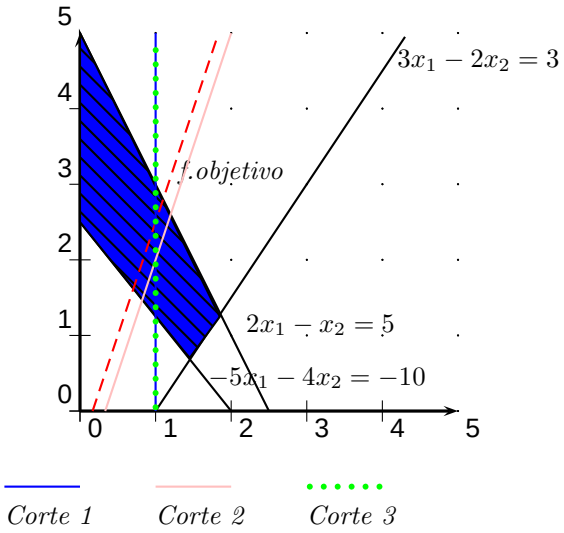


Figura 3.5: Gráfica Planos de Corte
Fuente:Elaboración propia.

3.2.2. Adaptaciones del Algoritmo Entero Mixto de Gomory

El algoritmo tradicional de Gomory genera un corte por iteración y añade éste a la formulación y resuelve. Por supuesto, una alternativa es generar varios cortes a partir de la solución básica actual, comúnmente el algoritmo genera un corte por cada variable 0-1 para la cual su valor es actualmente fraccional. Esta idea fue tomada de (Balas *et al.*, 1996*a*), en donde definen como un *round de cortes* a un conjunto de planos de corte generados para algunas o todas las variables fracionales binarias de la solución actual de la Relajación Lineal.

A continuación se mencionan algunos resultados importantes encontrados en (Balas *et al.*, 1996*a*), los cuales fueron aplicados al presente trabajo de investigación.

- Dada una cantidad de tiempo computacional, el mejor resultado en el valor de la función objetivo fue encontrado cuando los cortes fueron generados para todas las variables fracionales 0-1.
- Los cortes generados en cada nodo se continuan utilizando en todos aquellos nodos donde se mantienen las restricciones, es decir, para los nodos subsecuentes, pero no para aquellos que están en el mismo nivel de creación.
- Remover de la lista de restricciones actuales todos los cortes de Gomory generados en iteraciones previas, cuyas correspondientes variables de holgura son básicas en el punto fraccional actual. La Relajación Lineal resultante es resuelta nuevamente. Este proceso de *limpieza* en el Programa Lineal principal tiene al menos dos efectos positivos:

1. Reduce la inestabilidad numérica de la base, haciendo el cálculo de los cortes de Gomory más exactos.
2. Mantiene el Problema Lineal pequeño, por lo tanto se reduce el esfuerzo computacional ya que el LP es resuelto después de añadir los cortes. Más aún, estos planos de corte y ahora el LP pequeño, serán heredados por los hijos del nodo actual.

Una de las desventajas que presenta el uso del Algoritmo Entero Mixto de Gomory es que el número de desigualdades crece exponencialmente; por lo tanto, remover las restricciones de las variables básica disminuye el tamaño del problema y por lo tanto el tiempo de solución.

Este algoritmo tiene una convergencia finita bajo el supuesto de que la variable que define a la función objetivo debe ser entera. Sin esta suposición no se conoce un algoritmo de planos cortantes finito (Nemhauser y L., 1988). Esto conlleva a que las variables de decisión que conforman a la función objetivo deben de ser enteras. Dichas variables son las que representan los movimientos a la izquierda X_{ijN}, derecha X_{ijP}, arriba Y_{ijP} y abajo Y_{ijN} de los departamentos i y j, así que la distancia que deben desplazarse, para alcanzar el arreglo óptimo, debe ser entera.

Una vez que se han derivado desigualdades válidas para mejorar la formulación, mejorar en el sentido de ajustar la cubierta convexa, no es práctico añadir todas las desigualdades válidas que pueden ser generadas, es necesario realizar un preprocesamiento de restricciones, esto es, llevar un registro de las restricciones, ya que en ocasiones suelen repetirse, o bien, una es dominada por otra que se ha generado recientemente.

El hecho de expresar las desigualdades válidas en función de las variables de decisión originales, es para evitar el uso del Método Simplex Dual. Con el Método Dual la única solución factible y a la vez óptima, se obtiene cuando termina el algoritmo. Por lo tanto, si después de muchas iteracions se decide interrumpir el algoritmo, desde el punto de vista práctico no se ha conseguido nada, pues ni siquiera se tiene una solución factible, lo mejor que se puede lograr, es definir una cota de la función objetivo del problema (Prawda, 2004).

3.2.3. Algoritmo *Branch* & *Bound*

RAMIFICACIÓN. Definición de la regla de Ramificación. Se selecciona una variable binaria P_{ij} o Q_{ij} sobre la cual se ramificaran dos nuevos subproblemas, por un lado tendremos uno con la restrición $P_{ij} = 0$ y otro con $P_{ij} = 1$, la razón por la cual se ramifica con estos valores, es precisamente porque las variables binarias sólo pueden tomar el valor de 0 o bien de 1.

El criterio para escoger la variable binaria es optar por la "menos entera", es decir la variable más alejada de los dos enteros que la rodean. Este criterio fue el que resultó más efectivo al momento de resolver los ejemplos que se presentan en el capítulo siguiente. Además se realiza una Búsqueda en profundidad mientras que se realice la ramificación y después se utiliza la Mejor Cota después de podar el nodo.

ACOTACIÓN. Los nuevos subproblemas constituyen el árbol de ramificación o árbol de búsqueda, y se procede nuevamente a resolver la Relajación Lineal y a generar nuevos cortes.

SONDEO. Cada nuevo subproblema se elimina (sondea/poda) en cualquiera de los siguientes casos

1. El valor de la función objetivo es igual o peor que una solución entera conocida.
2. La solución del problema es infactible.
3. Satisface todas las resticciones de integralidad, el valor de la función objetivo es mejor que la solución actual, entonces esta solución se convierte en la actual y se aplica de nuevo el caso primero a todos los problemas no sondeados con el nuevo mejor valor de la función objetivo.

El algoritmo $B\,\&\,C$ consta de tres partes fundamentales

1. Heurística primal: Heurística encargada de generar una solución factible mejor que la que ya se conoce.
2. Rutina de Separación: Fase medular del algoritmo. Dada una solución fraccionaria, devolver una desigualdad no satisfecha por la solución fraccionaria pero *válida*, esto es, que las posibles soluciones enteras cumplan dicha restricción. Se intenta buscar restricciones violadas válidas, que son añadidas a la *RL*.
3. Ramificación: Entra la rutina del algoritmo *Branch & Bound*. Al ramificar se fijarán los valores de ciertas variables.

A continuación, se describen los principales componentes del algoritmo:

Heurística Primal

Definición de Cotas Cota Superior Z_{UB}: Se obtiene de la solución del *Problema de Asignación Lineal (LAP)*, dado que proporciona el arreglo de máxima dimensión, es decir, el peor caso. Esto es,

$$Maximizar \quad \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_{ij} \tag{3.15}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, 3, ..., n$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad j = 1, 2, 3, ..., n$$

$$x_{ij} \in [0, 1]$$

Cota Inferior Z_{LB}: Se obtiene de la solución de la Relajación Lineal (RL), a la que se irán añadiendo las restricciones obtenidas en el proceso de planos de corte.

El hecho de utilizar la RL en lugar de la Relajación de Lagrange, también utilizada en problemas de optimización combinatoria, es la siguiente,

Teorema 1 *El valor óptimo de la relajación lagrangiana de un problema de minimización es una cota inferior para él, mayor o igual que el valor óptimo de la relajación lineal. Además, si la relajación lineal es un problema cuya región factible es un poliedro entero, entonces ambos valores óptimos coinciden (Salazar, 2001).*

Hasta este momento sabemos que $Z_{UB} \geq Z^i* > Z_{LB}$, donde Z^i* es el valor óptimo de la función objetivo.

Rutina de Separación

Existen cortes de dos tipos (1) El corte nunca elimina soluciones enteras. Una vez generado, puede ser considerado para cualquier nodo (subproblema)del árbol. (2) El corte puede eliminar soluciones enteras que no son útiles para la minimización. Esta restricción será válida para un subproblema y para todos los sucesores que resulten del sub-árbol. Para los ejemplos realizados en el capítulo siguiente se generaron cortes del tipo 2, conocidos en la literatura como *locales* (Balas *et al.*, 1996*a*).

Criterio de paro. Considerando el efecto de los cortes agregados a la RL, si no se aprecia una mejora significativa en el valor de la función objetivo *v.f.o.*, se deja de generar nuevos cortes.

La integración de elementos que se presenta en este diagrama, tambien hace referencia a la integración de los conocimientos aportados por diferentes autores consultados.

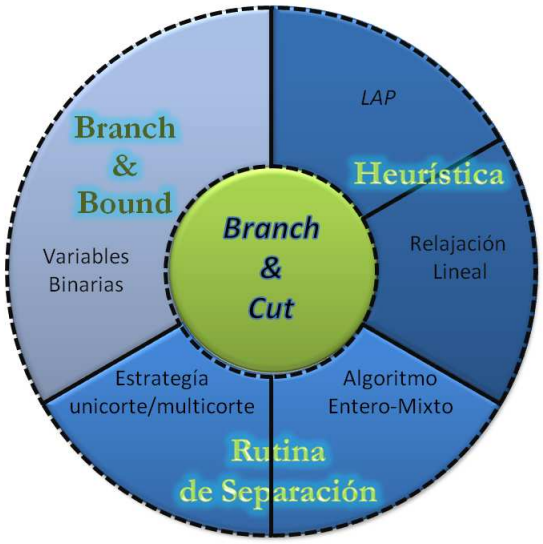


Figura 3.6: Integración de elementos para el diseño del algoritmo

Autor	Aportación
(Lawler y D., 1966)	Algoritmo B & B
(Mora, 2009)	Reglas de ramificación y sondeo
(Lawler y D., 1966)	Algoritmo B & B
(Heragu, 2006)	Problema de Asignación Lineal y formulación del problema
(Gennady, 2003)	Algoritmo de Planos de Corte
(Balas *et al.*, 1996*a*)	Adaptaciones al algoritmo de Planos de Corte
(Nemhauser y L., 1988)	Convergencia de Planos de Corte

Tabla 3.5: Integración de elementos del algoritmo propuesto

3.2.4. Complejidad del Algoritmo

Para cada nodo del árbol de búsqueda se deben resolver dos nuevos sub-problemas, ya que la ramificación se hace sobre alguna de las variables binarias. Para encontrar la solución óptima de cada *RL* del nodo, se recurre al Método Simplex, en este sentido la complejidad de esta sub-rutina del algoritmo es:

Teorema 2 *Para cada $n > 1$ existe un LP con $2n$ ecuaciones, $3n$ varibles, y coeficientes enteros con valor absoluto limitado por 4,tal que al método simplex le puede tomar $2^d - 1$ iteraciones encontrar el óptimo (Papadimitriou y K., 1998).*

Entonces, la complejidad aproximada del algoritmo propuesto queda expresada de la siguiente forma:

$$O\left(n\left(2^{n+1}-1\right)\right)$$

Finalmente se presenta el diagrama de flujo del algoritmo propuesto y en el siguiente capítulo se muestra el funcionamiento del algoritmo aplicado a algunos ejemplos.

PROCEDIMIENTO

Paso 1. Hallar una solución factible relativamente buena utilizando el Problema de Asignación Lineal. Conseguiremos una cota superior z_{UB} para nuestro problema.

Paso 2. Iniciar nodo raiz

Paso 3. A partir de resolver la Relajación Lineal RL obtendremos una cota inferior z_{LB}

Paso 4. Si es factible, entonces continuar al siguiente paso, en caso contrario, sondear el nodo.

Paso 5. Comparar el valor de la función objetivo con el valor anterior obtenido.

Paso 6. En caso de que el *v.f.o* no se mejore entonces, proceder a ramificar

Paso 7. Preguntar si la lista de variables binarias es vacía, es decir, si todas las variables $\in [0, 1]$

Paso 8. Utilizar el algoritmo de separación para regresar una desigualdad violada por la solución fraccional pero válida.

Paso 9. Se debe explorar todos los nodos de un mismo nivel antes de continuar.

Paso 10. Se satura cuando no se generan nodos a partir de él, debido a infactibilidad, se cumple la integralidad o tiene un *v.f.o* peor al anterior.

DIAGRAMA

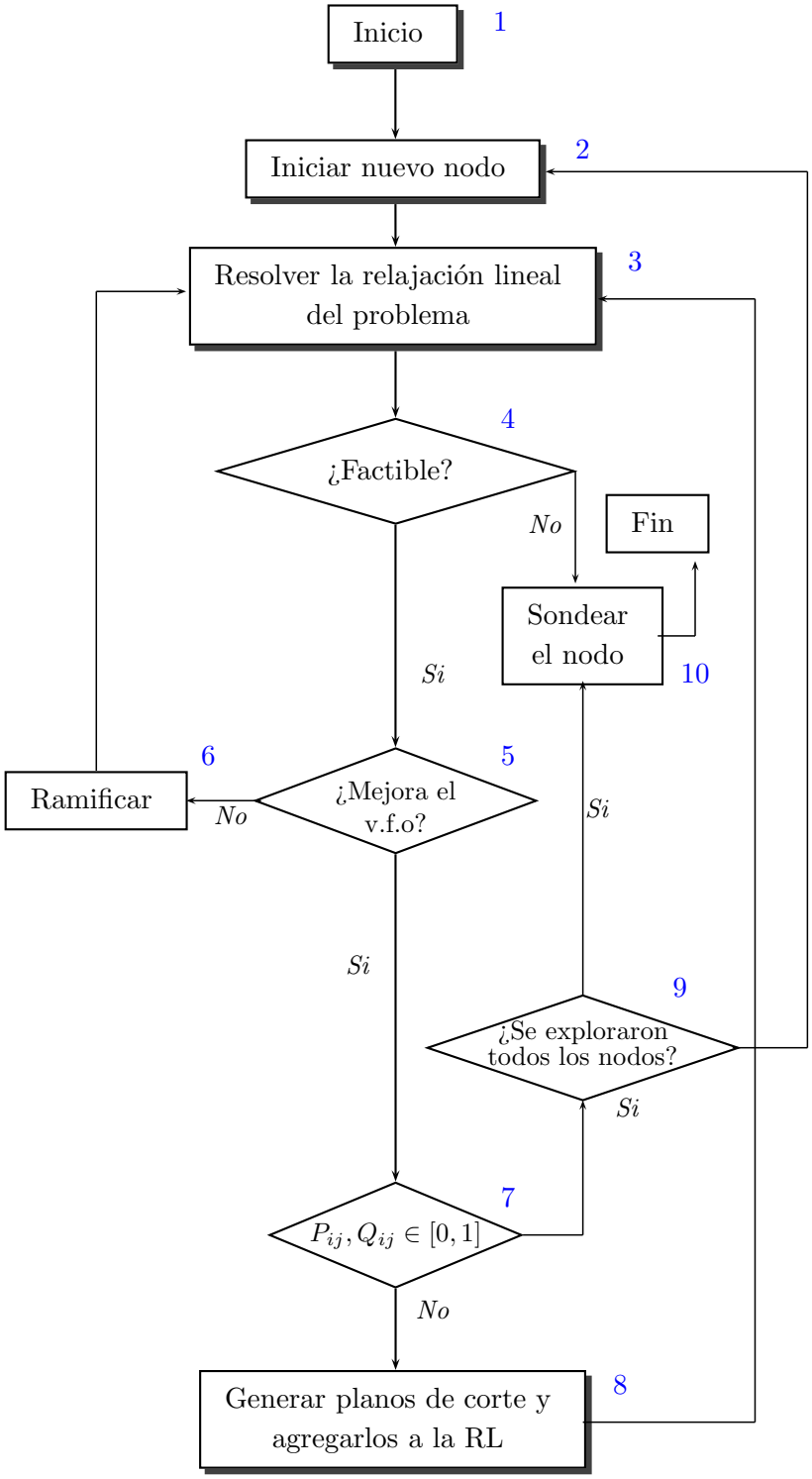


Figura 3.7: Diagrama de Flujo del Algoritmo Propuesto. Fuente: Elaboración propia

Capítulo 4

Aplicación

En esta sección se muestra cómo se aplica el algoritmo propuesto a ciertos ejemplos del problema de Distribución de Planta. Para entender el funcionamiento del algorimto, se desarrolla paso a paso para el caso en que se tienen 2 instalaciones, es decir, determinar cuál será el arreglo óptimo.

- Desarrollo del algoritmo con un problema de 2 instalaciones.

El modelo para este problema es:

$$
\begin{aligned}
& Minimizar \quad z = 10\left(X_{12P} + X_{12N} + Y_{12P} + Y_{12N}\right) \\
& sujeto \ \ a: \\
& X_1 - X_2 + MP_{12} + MQ_{12} \geq 25 && (4.1) \\
- \quad & X_1 + X_2 + MP_{12} + M\left(1 - Q_{12}\right) \geq 25 && (4.2) \\
& Y_1 - Y_2 + M\left(1 - P_{12}\right) + MQ_{12} \geq 20 && (4.3) \\
- \quad & Y_1 + Y_2 + M\left(1 - P_{12}\right) + M\left(1 - Q_{12}\right) \geq 20 && (4.4) \\
& X_1 - X_2 - X_{12P} + X_{12N} = 0 && (4.5) \\
& Y_1 - Y_2 - Y_{12P} + Y_{12N} = 0 && (4.6) \\
& X_1, X_2, X_{12P}, X_{12N}, Y_1, Y_2, Y_{12P}, Y_{12N} \geq 0
\end{aligned}
$$

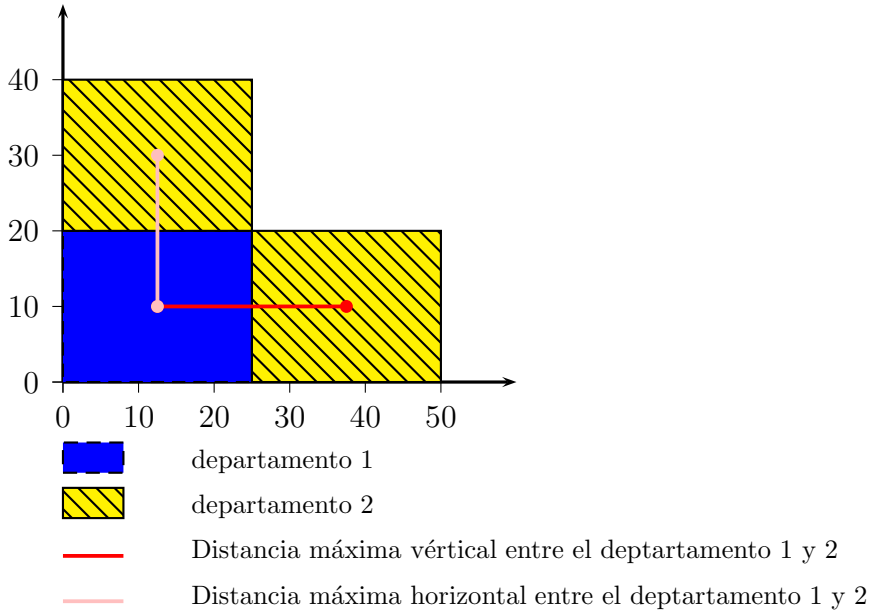
Oficina	1	2
1	-	10
2	10	-

Tabla 4.1: Matriz de traslados entre oficinas. Flujo que intercambian

Oficina	Largo	Ancho
1	25	20
2	25	20

Tabla 4.2: Dimensiones de las oficinas.

Paso 1. La cota superior z_{UB} la obtenemos con el arreglo lineal de las instalaciones *(LAP)*, en este caso será:



El departamento 2 puede estar en las coordenadas x, y $(15, 30)$ o bien, $(10, 40)$, la máxima distancia de separación entre departamentos es de 25 unidades, es decir, el centro de cada departamento puede estar separado hasta por un máximo de 25 unidades, por lo tanto, el intercambio de materiales se ve afectado por dicha distancia. Ahora sabemos que nuestra cota superior es 25 unidades.
En este caso particular, sólo hay dos combinaciones reales posibles de la disposición de las instalaciones, podemos decir que son cuatro posibilidades, pero esto es porque tenemos dos en cada eje y corresponden una con otra.

Paso 2. Se inicia el nodo 0.

Paso 3. Resolviendo la Relajación Lineal RL, se tiene la siguiente tabla óptima.

Paso 4. Dado que el problema es factible continuamos con el siguiente paso Paso 5. Para este punto inicial, no tenemos una cota de referencia con la cual comparar si ha mejorado o no el valor de la función objetivo, por lo tanto seguimos adelante. Paso 6. En este problema, tenemos dos variables que deben ser binarias, P_{12} y Q_{12}, de la solución de la relajación lineal, tenamos que $P_{12} = 0.25$ y $Q_{12} = 0$, entonces $Q_{12} \in [0, 1]$ pero P_{12} es

Base	X_{12P}	X_{12N}	Y_{12P}	Y_{12N}	X_1	X_2	P_{12}	Q_{12}	Y_1	Y_2	S_1	S_2	S_3	RHS
S_1	0.0	0.0	0.0	0.0	2.0	-2.0	0.0	200	0.0	0.0	1.0	0.0	0.0	100.0
S_2	0.0	0.0	0.0	0.0	-1.0	1.0	0.0	-200	-1.0	1.0	0.0	1.0	0.0	55.0
S_3	0.0	0.0	0.0	0.0	-1.0	1.0	0.0	0.0	1.0	-1.0	0.0	0.0	1.0	155.0
P_{12}	0.0	0.0	0.0	0.0	0.010	-0.010	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.25
X_2	-1.0	1.0	0.0	0.0	1.0	-1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
X_4	0.0	0.0	-1.0	1.0	0.0	0.0	0.0	0.0	1.0	-1.0	0.0	0.0	0.0	0.0
z	10.0	10.0	10.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabla 4.3: Tabla óptima 1 del Problema 1

fraccional, por lo tanto elegimos esta variable y continuamos. Paso 7. Generamos el plano de corte asociado a la variable P_{12} y lo agregamos a la relajación lineal del problema. Regresamos al Paso 3.

El plano de corte obtenido en este punto es:

$$0.3433P_{12} + 0.3366Q_{12} + 0.003333X_1 - 0.003333X_2 - 0.3333 \geq 0 \qquad \text{(Corte 1)}$$

La RL del problema ahora es:

$$\begin{aligned}
& Minimizar \quad z = 10\left(X_{12P} + X_{12N} + Y_{12P} + Y_{12N}\right) \\
& sujeto \ \ a: \\
& X_1 - X_2 + MP_{12} + MQ_{12} \geq 25 && (4.7) \\
- \quad & X_1 + X_2 + MP_{12} + M\left(1 - Q_{12}\right) \geq 25 && (4.8) \\
& Y_1 - Y_2 + M\left(1 - P_{12}\right) + MQ_{12} \geq 20 && (4.9) \\
- \quad & Y_1 + Y_2 + M\left(1 - P_{12}\right) + M\left(1 - Q_{12}\right) \geq 20 && (4.10) \\
& X_1 - X_2 - X_{12P} + X_{12N} = 0 && (4.11) \\
& Y_1 - Y_2 - Y_{12P} + Y_{12N} = 0 && (4.12) \\
& 0.0033X_1 - 0.0033X_2 + 0.3433P_{12} + 0.3366Q_{12} \geq 0.3333 && (4.13) \\
& X_1, X_2, X_{12P}, X_{12N}, Y_1, Y_2, Y_{12P}, Y_{12N} \geq 0
\end{aligned}$$

Nuevamente resolvemos RL como lo indica el Paso 3 y la tabla obtenida es:

Paso 4. El problema es factible. Paso 5. El valor de la función objetivo es igual al valor del obtenido anteriormente. Paso 6. $P_{12} = 0.886$ y $Q_{12} = 0.086$, ambos son fraccionales. Paso 7. Se deducen 2 nuevas restriciones y las agregamos a la relajación lineal del problema. Regresamos al Paso 3.

Base	X_{12P}	X_{12N}	Y_{12P}	Y_{12N}	X_1	X_2	P_{12}	Q_{12}	Y_1	Y_2	S_1	S_2	S_3	S_4	RHS
S_1	1.000	-1.000	-1.000	1.000	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	155.00
S_4	-0.029	0.029	0.010	-0.010	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.010	0.0	1.0	72.25
S_3	-0.971	0.971	0.990	-0.990	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.010	1.0	0.0	2.74
P_{12}	0.005	-0.005	-0.005	0.005	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.005	0.0	0.0	0.886
Q_{12}	0.005	-0.005	0.005	-0.005	0.0	0.0	0.0	1.0	0.0	0.0	0.0	-0.005	0.0	0.0	0.086
X_1	-1.000	1.000	0.000	0.000	1.0	-1.0	0.0	0.0	0.0	0.0	0.0	0.000	0.0	0.0	0.000
Y_1	0.000	0.000	-1.000	1.000	0.0	0.0	0.0	0.0	1.0	-1.0	0.0	0.000	0.0	0.0	0.000
z	10.000	10.000	10.000	10.000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.000	0.0	0.0	0.0

Tabla 4.4: Tabla óptima 2 del problema 1

Los planos de corte obtenidos son:

$$0.005X_{12P} + 0.03886X_{12N} + 0.03886Y_{12P} + 0.005Y_{12N} + 4.4248P_{12} + 3.3482Q_{12} + 0.033108X_1 - 0.033105X_2 - 4.3215 \geq 0$$

(Corte 2)

$$0.005X_{12P} + 0.00047X_{12N} + 0.005Y_{12P} + 0.00047Y_{12N} + 0.094562P_{12} - 0.000458Q_{12} - 0.00000914X_1 + 0.00000914X_2 - 0.096848 \geq 0$$

(Corte 3)

La solución de este nuevo problema sigue siendo $z = 0$. Continuamos con el ciclo de generación de cortes. Se generan 4 nuevos cortes, es decir 2 ciclos más, en donde la solución mejora $z = 0$. Comenzamos un nuevo ciclo, agregamos 2 cortes más, pero la solucón sigue con el mismo valor en la función objetivo, por lo que se procede con el Paso 6. Ramificar.

Arbitrariamente se decide sobre que variable se ramificará, en este caso se escogió la variable P_{12}. Sobre esta rama se realizan 3 cortes más para el caso en donde $P_{12} = 1$ y en donde $P_{12} = 0$ se obtiene un valor de $Q_{12} = 25.25$ y $z = 47,949.91$.

Se continua con el procedimiento y se obtiene el siguiente árbol de búsqueda.

De los dos árboles de búsqueda anteriores, puede observarse que se redujo en iteraciones la búsqueda del óptimo con el algoritmo B & C en comparación con B & B. Este ejemplo es puramente ilustrativo, lo que pretende es ayudar a entender el funcionamiento del algoritmo y comprender como sucede la reducción de iteraciones.

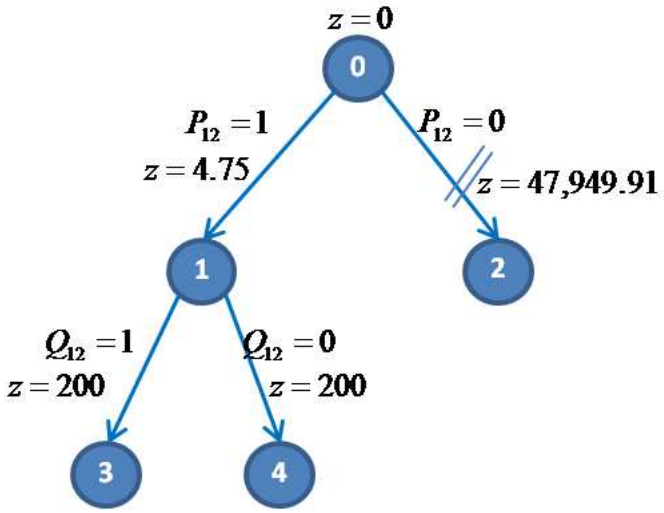


Figura 4.1: Árbol de Búsqueda para 2 instalaciones B & C

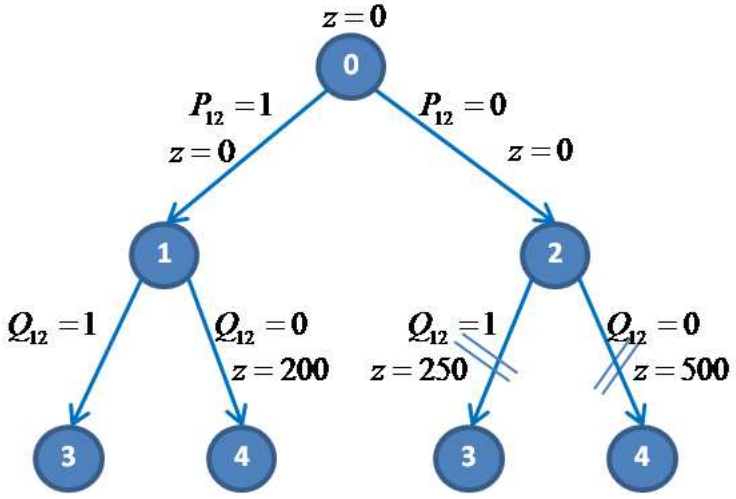


Figura 4.2: Árbol de Búsqueda para 2 instalaciones B & B

4.1. B & C en un problema de 3 instalaciones

Para una distribución de departamentos que involucra 3 instalaciones, se tienen 20 restricciones, a continuación se presentan los datos necesarios para contruir el modelo.

Oficina	1	2	3
1	-	10	15
2	10	-	30
3	15	30	-

Tabla 4.5: Matriz de traslados entre oficinas. Flujo que intercambian

Oficina	Largo	Ancho
1	25	20
2	25	20
3	35	30

Tabla 4.6: Dimensiones de las oficinas.

La siguiente tabla muestra algunos de los resultados para los nodos del árbol.

Nodo	*v.f.o* inicial	cortes generados	*v.f.o* inicial
Nodo 0	0	11	68.7371
Nodo 1	369.6802	2	749.64
Nodo 2	24.8172	4	680.7322
Nodo 3	750.61	2	750.61
Nodo 4	750.61	2	750.61
Nodo 6	24.8172	2	24.8172
Nodo 7	750.61	2	837.44
Nodo 8	750.61	2	750.61
Nodo 15	860.85	2	860.85
Nodo 16	1342	2	1774.63
Nodo 17	750.61	2	837.44
Nodo 18	1262.17	2	1350
Nodo 27	1125.26	7	1750.6
Nodo 28	1625.23	2	**1625.4**

Tabla 4.7: Tabla de resultados.

El valor óptimo es $z = 1625$. Si el problema se resolviera utilizando el algoritmo B & B.

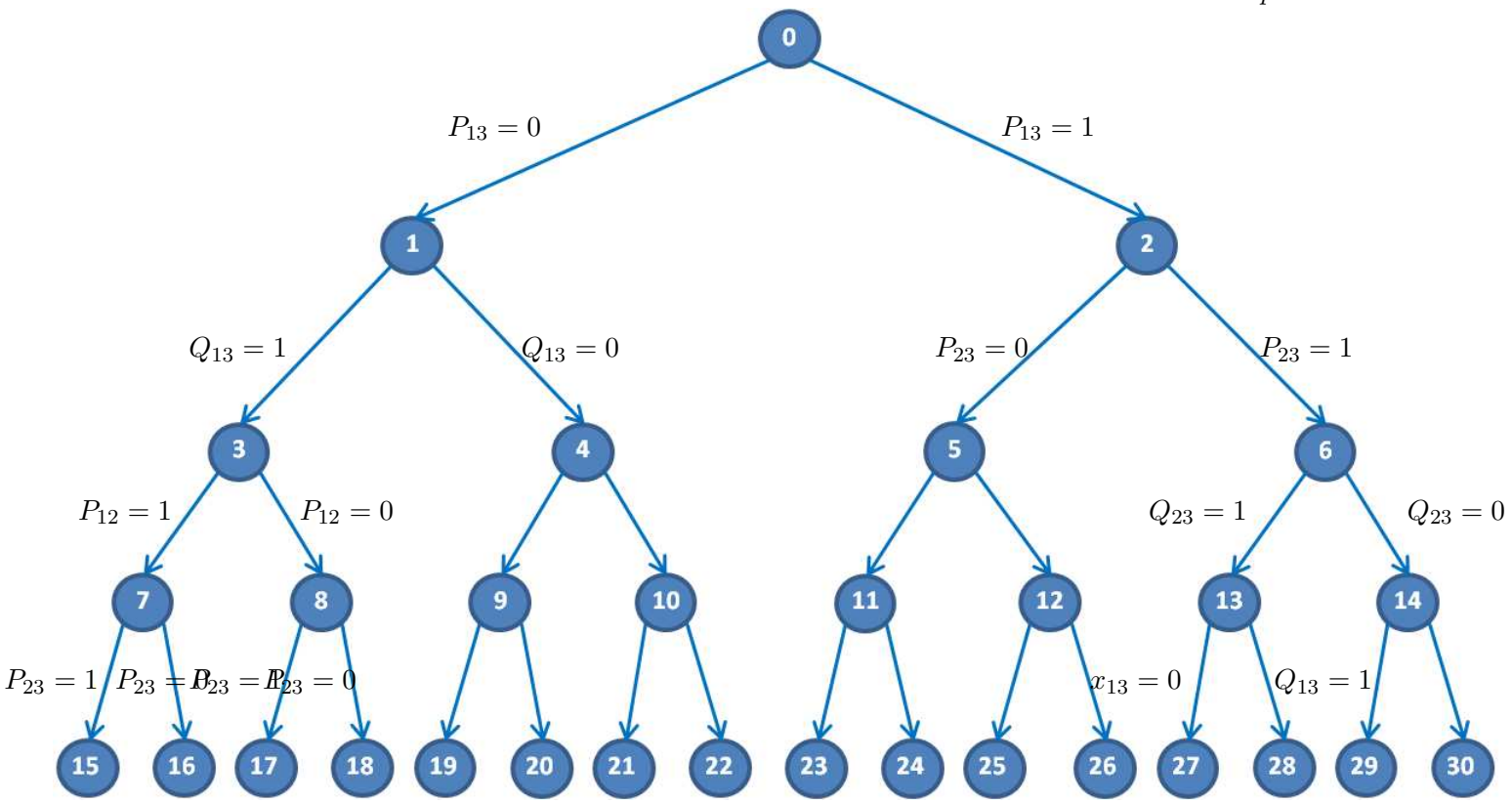


Figura 4.3: Árbol de Búsqueda para 3 instalaciones B & C

En conclusión, resolviendo con:

- Branch & Bound se tienen 128 nodos.
- B & C se tienen 30 nodos, 5 iteraciones (en algunos nodos se genera sólo un corte y en otros hasta un máximo de 11). Esto es con la estrategia de generar un sólo corte en cada iteración, es decir, un corte sólo una variable binaria.

Ahora bien, utilizando la estrategia multicorte antes mencionada, se reduce en una iteración encontrar la solución del problema, esto es 14 nodos. A continuación se presenta el árbol de búsqueda final para este problema.

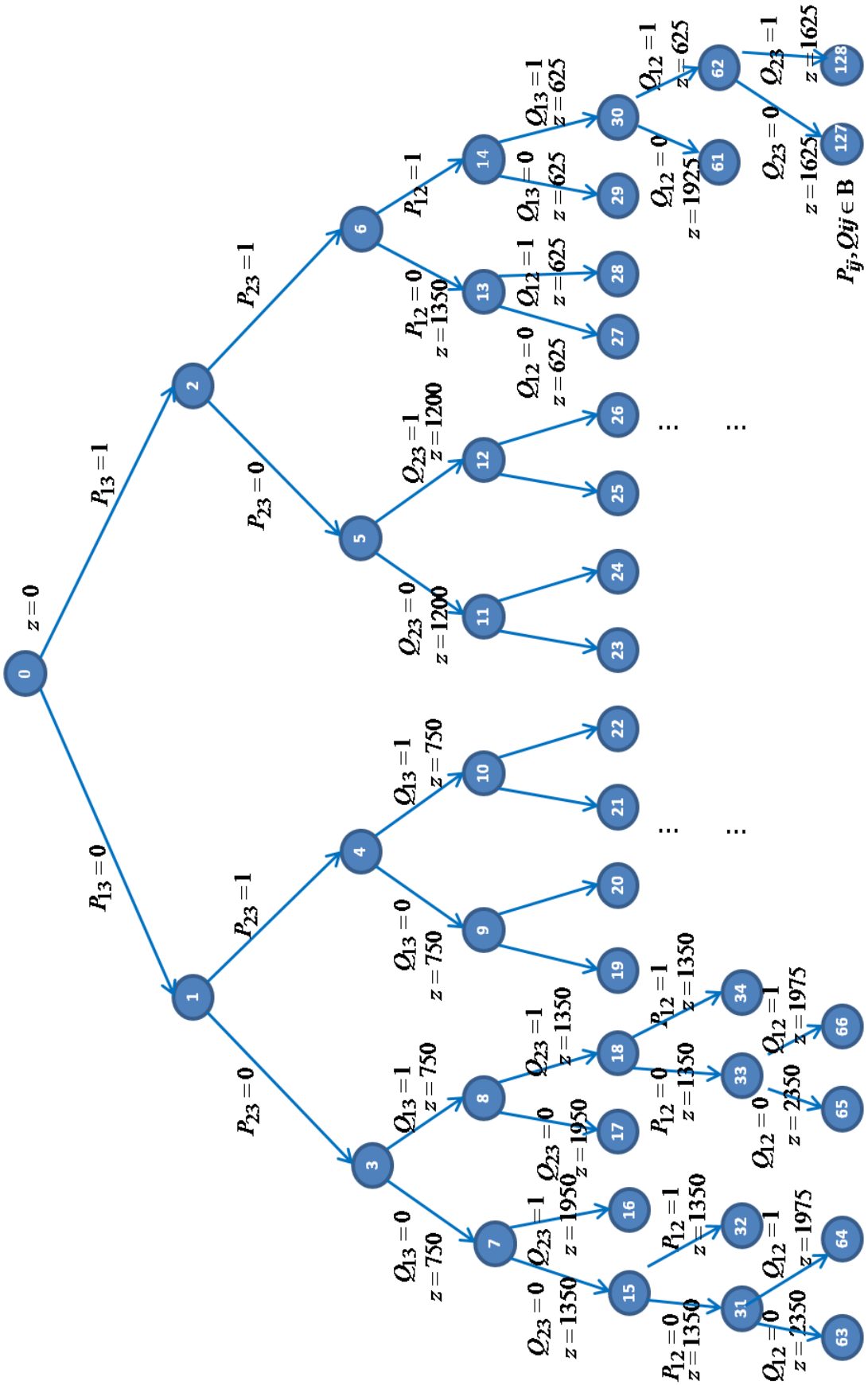


Figura 4.4: Árbol de Búsqueda para 3 instalaciones B & B

4.2. B & C en un problema de 5 instalaciones

El ejemplo de aplicación en un problema de tamaño pequeño (5 departamentos) ha sido tomado de Heragu (2006). El cual se describe a continuación.

Una compañía de seguros quiere determinar la localización de 5 cubículos/oficinas de diferentes dimensiones en el último piso del edificio de su oficina central. Dichas oficinas no tienen ninguna interacción con oficinas de otros pisos. Recientemente se llevó a cabo un muestreo de la forma de trabajo y en un periodo de 12 meses se estimó el número de traslados realizados por el personal entre estas 5 oficinas. Los traslados estimados y las dimensiones de cada oficina se muestran en las siguientes tablas.

Oficina	1	2	3	4	5
1	-	10	15	20	0
2	10	-	30	35	10
3	15	30	-	10	20
4	20	35	10	-	15
5	0	10	20	15	-

Tabla 4.8: Matriz de traslados entre oficinas. Flujo que intercambian

Oficina	Largo	Ancho
1	25	20
2	25	20
3	35	30
4	30	20
5	35	20

Tabla 4.9: Dimensiones de las oficinas.

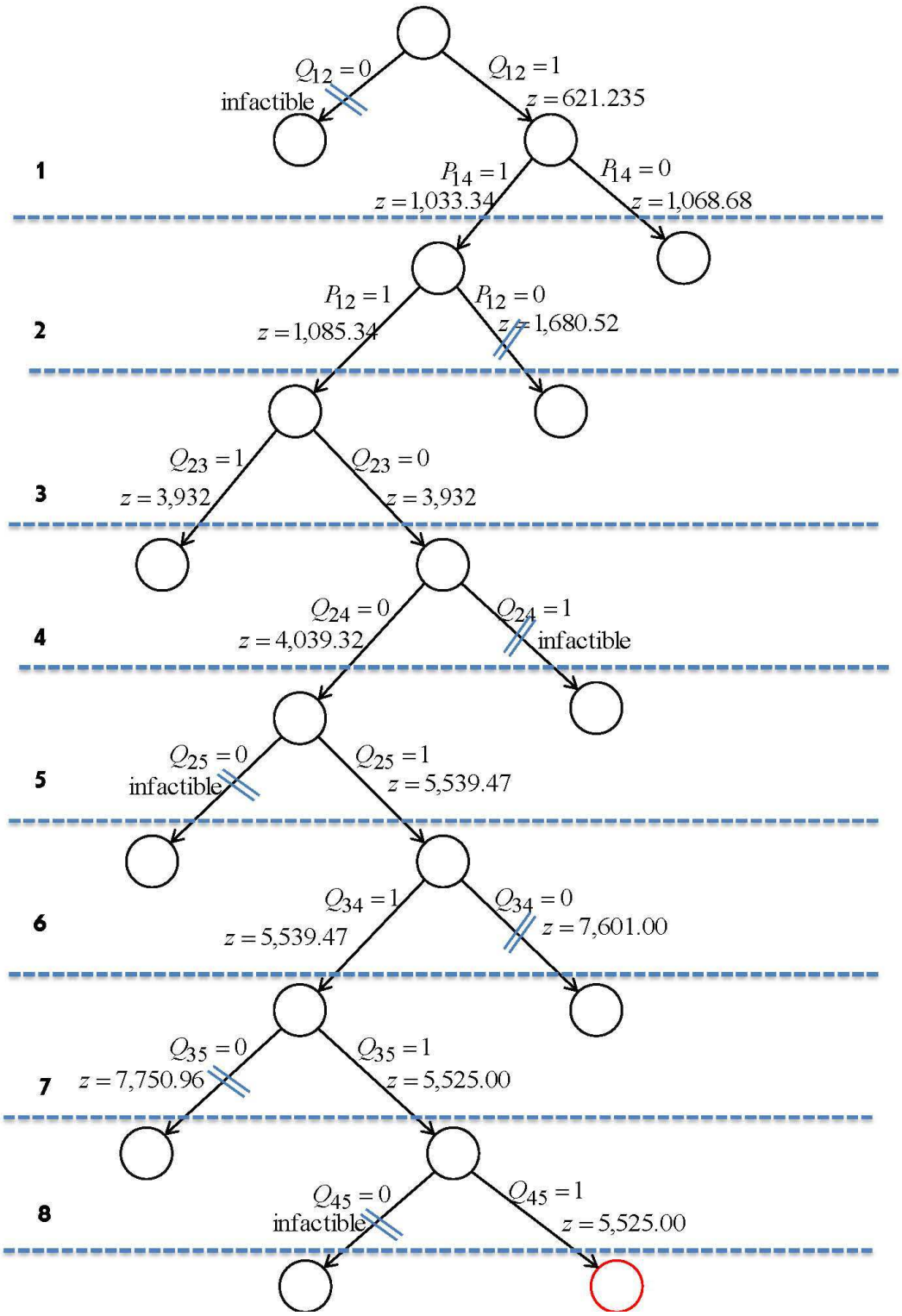


Figura 4.5: Árbol de Búsqueda para 5 instalaciones B & C.

El valor óptimo es el que se encuentra al final del árbol, en el nodo de color rojo. En este ejemplo varios de los subproblemas resultaron infactibles lo que redujo en gran medida el árbol de búsqueda.

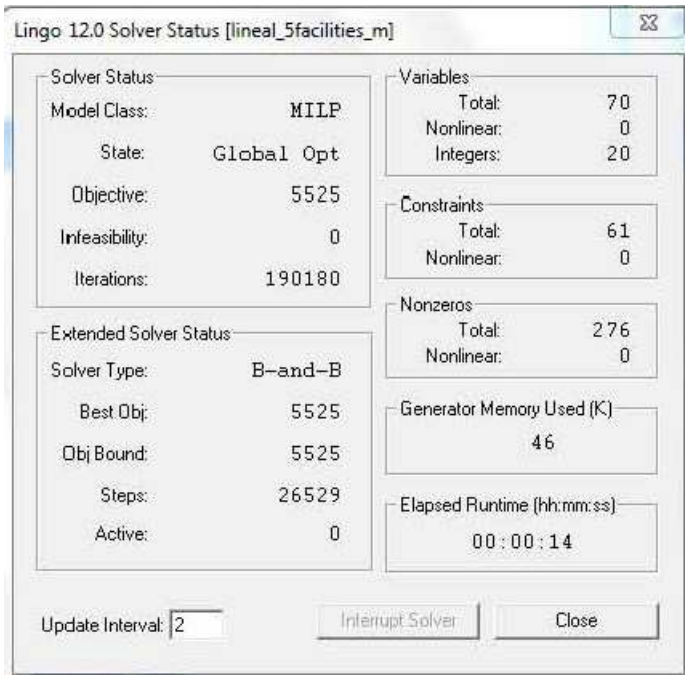


Figura 4.6: Reporte de Lingo 5 instalaciones.

4.2.1. Análisis de la operación del Algoritmo

Para implementar exitosamente el algoritmo se enfrentaron con los siguientes problemas.

Formulación
Una de las principales características del problema *FLP* es que al resolver la RL, el valor de la función objetivo es cero y dado que se trata de un problema de minimización este valor no se puede mejorar. Para salir de este óptimo local, inmediatamente se recurrió a los Planos de corte, con el fin de encontrar el valor para alguna variable binaria y activiar las restricciones que impiden la sobreposición de instalaciones.

El valor de M en la formulación es el que ayuda a mantener las restricciones que impiden el traslape de departamentos. Para ciertos valores de M se tiene una tabla óptima unimodular y aunque se probaron diferentes valores para dicho parámetro, sólo algunos valores cambiaron a fraccionales, pero su valor era demasiado pequeño (del orden de 0.001 y 0.005) como para obtener un plano cortante que ayudará a reforzar a la solución. Por esta razón se decidió utilizar el Algorimo Entero Mixto de Gomory, ya que en la implementación el algorimto fraccional no resultaba adecuado.

Obtención y manejo de la información
Inicialmente se trabajó con el algoritmo de forma manual. Los cortes se obtuvieron uno por uno y se fueron incorporando a la *RL*, la cual se resolvió por medio de software comercial (Lingo 11 y WinQsb 2.0). Esto se realizó en cada nodo del árbol de búsqueda, hasta encontrar la solución óptima. Resolver un ejemplo que involucre tres instalaciones requirió de un trabajo exhaustivo.

Por lo anterior, se decidió programar la rutina de obtención de corte, ya que esto facilitó en cierta medida el trabajo.

En cada iteración se requiere de la tabla óptima del problema. Los programas comerciales que se utilizan para obtener la solución de la *RL* no brindan información detallada sobre ésta, al menos no para los casos que involucran más de cuatro instalaciones. Por lo tanto, fue necesario programar el Método Simplex en MATLab y con esto, tener la información completa sobre la tabla óptima. En este sentido se programó el Método Simplex de las Dos Fases y se validó su funcionamiento con diversos ejemplos para los cuales ya se conocía la solución óptima.

Con la rutina para obtener el plano de corte y la tabla óptima del Simplex se logró implementar el algoritmo para el ejemplo que involucra cinco instalaciones.

Cabe mencionar que no se cuenta con un programa que conjunte todo el funcionamiento del algoritmo, ya que el objetivo principal no es contar con un programa, si no con un algoritmo.

Conclusiones y Extensiones

El presente trabajo de investigación conjunta conceptos de la ingeniería industrial y de optimización matemática. Por un lado se pretende dar una opción a todas aquellas empresas que requieren realizar una distribución de departamentos y/o maquinaria con el fin de minimizar costo de manejo de materiales. El área de investigación de operaciones aborda los problemas reales desde una perspectiva matemática, es decir, proponer un modelo que se asemeje a la realidad con el fin de optimizar los procesos, o bien, eventos.

La propuesta de algoritmo presentada puede resultar eficiente para afrontar problemas de distribución de planta que involucran una gran cantidad de variables, es decir, en donde se requiere que una gran cantidad de elementos/departamentos sean ubicados en una instalación.

Se logra obtener la solución óptima del problema con el algoritmo propuesto.

Se cumplió con el objetivo principal de diseñar un algoritmo Branch & Cut (B & C) para resolver el problema de distribución de planta bidimensional con áreas desiguales, ya que se obtuvo el arreglo óptimo para los ejemplos presentados.

Se observa que reduce el número de iteraciones comparando los algoritmos B & C y B & B para los ejemplos aquí desarrollados. Se observa que conforme crece el tamaño de entrada la reducción de iteraciones es más significativa.

Cabe mencionar que una buena propuesta de trabajo futuro será implementar todo el algoritmo en un sólo programa, ya que, hasta este momento se obtienen por separado la tabla óptima del simplex del problema lineal y los cortes asociados a dicha relajación.

Existen varios aspectos en los que aún se puede mejorar la propuesta de algoritmo que aquí se desarrolla. A continuación se enumeran algunos de éstos.

- *Lift and Project Cuts* (Levantamiento y Proyección de Cortes). Al incorporar este tipo de planos cortantes al algoritmo propuesto se tendrán cortes globalmente válidos para todo el árbol de búsqueda. Éstos han mostrado ser una forma efectiva de reforzar la formulación de los problemas *MIP* 0-1, no sólo son numéricamente más estables que los cortes enteros mixtos de Gomory, si no que para cada corte de Gomory existe un *lift and proyect cut* que lo domina (Balas *et al.*, 1996*b*).

- Incorporación de todos los elementos del algoritmo B & C propuesto, en un sólo programa.
- Implementar el algoritmo en ejemplos prototipo ampliamente estudiados y con esto comprobar su generalidad, como los presentados en Solimanpur y Jafari (2008) y Amaral (2008)

Bibliografía

Amaral, André (2008). n exact approach to the one-dimensional facility layout problem.. *Operations Research.* **56**, 1026–1033.

Balas, E., S. Ceria, G. Cornuéjols y N. Natraj (1996*a*). Gomory cuts revised. *Operations Research Letters* **19**, 1–9.

Balas, E., S. Ceria y G. Cornuéjols (1996*b*). Mixed 0-1 programming by lift-and-project in a branch and cut framework. *Management Science* **42**, 1229 – 1246.

Bazaraa, M.S. (1975). Computrized layour design: A branch and bound approach. *AIIE Transactions* **7**, 432–437.

Bixby, R.E. y E.K Lee (1998). Solving a truck dispatching scheduling problem using branch-and-cut. *Operations Research* **46**, 355–367.

Caprara, A. y Salazar J.J (1996). A branch-and-cut algorithm for a generalization of the uncapacitated facility location problem. *TOP* **4**, 135–163.

Dell´Amico, M., F. Maffioli y S. Martello (1997). *Annotated Bibliographies in Combinatorial Optimization.*. Wiley Interscience.

Dorf, R. y Kusiak A. (1994). *Handbook of Design, Manufacturing and Automation.*. John Wiley and Sons. Inc.

Flores, I. (2003). *Apuntes de Programación Entera*. UNAM. Facultad de Ingeniería.

Gennady, Shmonin (2003). Cutting planes, gomory mixed-integer general cutting planes.

Graves, Robert L. y Philip Wolfe (1953). *Recent Advances in Mathematical Programming.*. McGraw-Hill.

Heragu, S. y Alfa A. (1992). Experimental analysis of simulated annealing based algorithms for the layout problem.. *European Journal of Operational Research.* **57**, 190–202.

Heragu, Sunderesh S. (2006). *Facilities Design*. iUniverse, Inc.

Honiden, T. (2004). Tree structure modeling and genetic algorithm-based approach to unequal-area facility layout problem.. *IEMS* **3**, 123–128.

Koopmans, T.C. y M. Beckmann (1957). Assignment problems and the location of economic activities. *Journal of the Econometric Sciety* **25**, 53 – 76.

Kusiak, A. y Heragu S.S. (1987). The facility layout problem.. *European Journal of Operational Research North Holland* **29**, 229–251.

Lawler, E. y Wood D. (1966). Branch-and-bound methods a survey.. *Operations Research.* **14**, 699–719.

Love, R. y J. Wong (1976). Solving quadratic assignment problems with rectangular distances and integer programming.. *Naval Research Logistics Quarterly* **23**, 623–627.

Meller, R., V. Narayanan y P. Vance (1998). Optimal facility layout design.. *Operations Research Letters.* **23**, 117–127.

Mir, M. y Imam M. H. (2001). A hybrid optimization approach for layout design for layout design of unequal area facilies.. *Computers & Industrial Engineering.* **39**, 49–63.

Mora, H. (2009). *Temas de Optimización*. Universidad Nacional de Colombia. Facultad de Ciencias.

Neapolitan, R. y Naimipour K. (2009). *Foundations of Algorithms.*. Jones and Bartlett Publishers.

Nemhauser, G. y Wolsey L. (1988). *Integer and Combinatorial Optimization.*. Wiley.

Padberg, M. y Rinaldi G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems.. *SIAM Review* **33**, 60–100.

Papadimitriou, C. y Steiglitz K. (1998). *Combinatorial Optimization. Algorithms and Complexity*. Dover Publicatons, Inc.

Perron, S., P. Hansen, S. Le Digabel y N. Mladenovic (2010). Exact and heuristic solutions of the global supply chain problem with transfer pricing. *European Journal of Operations Research* **202**, 864 – 879.

Prawda, Juan (2004). *Métodos y Modelos de Investigación de Operaciones I y II.*. Limusa Noriega Editores.

Sahni, S. y T. González (1976). P-complete approximation problem. *Jopurnal of Associated Computing Machinery* **23**, 555–565.

Salazar, J. González (2001). *Programación Matemática*. Ediciones Diaz de Santos.

Scholz, D., A. Petrick y W. Domschke (2009). Stats: A slicing tree and tabu search based heuristic for the unequal area facility layout problem.. *European Journal of Operational Research.* **197**, 166–178.

Sherali, H., B. Fraticelli y R. Meller (2003). Enhanced model formulations for optimal facility layout.. *Operations Research.* **51**, 629–644.

Solimanpur, M. y A. Jafari (2008). Optimal solution for the two-dimensional facility layout problem using a branch-and-bound algorithm.. *Computers & Industrial Engineering.* **55**, 606–619.

Stecco, G., J.F Cordeau y E. Moretti (2008). A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. *Computers & Operations Research* **35**, 2635–2655.

Tavakkoli-Moghaddam, R., N. Javadian, B. Javadi y N. Safaei (2007). Design of a facility layout problema in celular manufacturing systems with stochastic demands.. *Applied Mathematics and Computation.* **184**, 721–728.

Wong, K. Y. (2010). Applying ant system for solving unequal area facility layout problems. *European Journal of Operational Research* **202**, 730 – 746.

Anexos

Programas

A continuación se presenta el programa con el cual se obtuvieron los cortes de Gomory para los problemas antes presentados.

```
clc
clear all

V1 = [1.000     0.000     0.066     0.207     0.056     0.151
0.800];  % VECTOR ASOCIADO A LA FILA DE LA VARIABLE
         % BINARIA DE LA TABLA FINAL DEL SIMPLEX SOBRE LA CUAL
         % SE REALIZARÁ EL CORTE

N = max(size(V1-2));

V2 = [0     0   0   0   0   0   1   1   1   1   1    0   0   0    0];
% VECTOR QUE INDICA SI UNA VARIABLE ES ENTERA O NO
% 1 - ENTERA, % 0 -- BINARIA

%%%%%%%%%%%%%%%%%%%% ALGORITMO DE CORTE%%%%%%%%%%%%%%%%%%%%
V1f = V1 - floor(V1);

h = V1(N) - floor(V1(N));

R1 = zeros(1,N);

for i=1:N-1

   if ((V2(i) == 1) && (V1f(i) <= h))

       R1(i) = V1f(i);

   elseif((V2(i) == 1) && (V1f(i) > h))

        R1(i) = (h/(1-h))*(1 - V1f(i));

   elseif((V2(i) == 0) && (V1(i) >= 0))

       R1(i) = V1(i);

   else

       R1(i) = (h/(1-h))*abs(V1(i));
```

```
   end
end
R1 = double(R1');


syms  X1   X2   X3    X4     X5    X6    X7      X8    X9     R;
% DEFINICIÓN DE VARIABLES, SLACKS Y SURPLUS.

%%%%HOLGURAS DE LAS RESTRICCIONES <= (PROPIAS DE LA FORMUALCIÓN)%%%%%%%%%
S1 = 75 + 100*X41 - 100*X51 - X61 + X62;
S2 = 70 + 100*X42 - 100*X52 - X61 + X63;
% DECLARACIÓN DE LAS VARIABLES DE HOLGURA Y SLACKS
% DE LAS RESTRICCIONES. ESTO PARA GENERAR UNA RESTRICCIÓN
% EN FUNCIÓN DE LAS VARIABLES ORIGINALES Y QUE NO SEA NECESARIO
% UTILIZAR EL MÉTODO SIMPLEX
S3 = 72.5 -X61 + X64 + 100*X43 - 100*X53;
S4 = 70 - X61 + X65 + 100*X44 - 100*X54;
S5 = 70 - X62 + X63 + 100*X45 - 100*X55;
S6 = 72.5 - X62 + X64 + 100*X46 - 100*X56;
S7 = 70 - X62 + X65 + 100*X47 - 100*X57;
S8 = 67.5 - X63 + X64 + 100*X48 - 100*X56;
S9 = 65 - X63 + X65 + 100*X49 - 100*X59;
S10 = 67.5 - X64 + X65 + 100*X50 - 100*X57;
S11 = 80 + X66 - X67 - 100*X41 + 100*X51;
S12 = 180 - X66 + X67 - 100*X41 - 100*X51;
.
.
.
%%%%%%%%AQUI SE VAN AGREGANDO LAS NUEVAS RESTRICCIONES%%%%HOLGURAS DE LAS
RESTRICCIONES <= "CORTES"%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

S31 = .16062e-1*X12+.22255e-1*X11-.31967e-2*X65+189/30500*X32
-.2100e-1*X59+1/200*X9+1/500*X31+1/1000*X33+1/500*X69+561/61000*X70
+189/12200*X10-.30000e-2*X63-.81967e-
2*X67+3/1000*X30+1.5722+189/61000*X34
-.94298*X56+.74298e-2*X64+3/1000*X18+567/61000*X6+2.1554*X46+3/1000*X5
-.30000e-2*X68-.1233e-2*X62+567/61000*X29+11/1000*X17;
S32 = .34202e-3*X12+.78215+.26300e-2*X11+1/500*X65+1/200*X66-.37331*X59
+7603705233327165/737869762948382064 64*X9+1/500*X33+1/500*X70+.50000*X52
+1/500*X10-.50000*X42-.20000e-
2*X63+7/1000*X30+2376157885414739/9223372036854775808*X21
+1/200*X22+7603705233327165/737869762948382064 64*X34+.23897e-1*X56-
.23897e-3*X64
+7/1000*X18+6/116449*X6+.23897e-1*X46+1/1000*X5-.70000e-2*X68+.23897e-
3*X62+37/102586*X29+1/1000*X17;
.
.
.

%%%%HOLGURAS DE LAS RESTRICCIONES >= (PROPIAS DEL LA FORMUALCIÓN%%%%%%%%
S49 = -25 + 100*X41 + 100*X51 + X61 - X62;
S50 = -30 + 100*X42 + 100*X52 + X61 - X63;
S51 = -27.5 + X61 - X64 + 100*X43 + 100*X53;
S52 = -30 + X61 - X65 + 100*X44 + 100*X54;
S53 = -30 + X62 - X63 + 100*X45 + 100*X55;
```

```
S54 = -27.5 + X62 - X64 + 100*X46 + 100*X56;
S55 = -30 + X62 - X65 + 100*X47 + 100*X57;
S56 = -32.5 + X63 - X64 + 100*X48 + 100*X56;
S57 = -35 + X63 - X65 + 100*X49 + 100*X59;
S58 = -32.5 + X64 - X65 + 100*X50 + 100*X57;
.
.
.

%AQUÍ SE VAN AGREGANDO LAS NUEVAS RESTRICIONES%%%%%%%%%%%%%%%HOLGURAS DE
LAS RESTRICCIONES >= "CORTES"%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
S = 0.3333*X41 + 0.3333*X51 + 1/75*X61 - 0.3333;
S60 = 1/70*X61 + 0.4286*X42 + 0.4286*X52 - 0.4286;
S61 = 0.3793*X53 + 0.3793*X43 + 2/145*X61 - 0.3793;
S62 = 3/700*X64-.42857e-2*X65+1/70*X61+.42857*X54-.42857+.42857*X44;
S63 = 1/70*X62 + 0.4286*X55 + 0.4286*X45 - 0.4286;
S64 = 2/145*X62 + 0.3793*X56 + 0.3793*X46 - 0.3793;
S65 = 1/70*X62 + 0.4286*X57 + 0.4286*X47 - 0.4286;
.
.
.

X = [X1   X2   X3   . . .  S1 S2 S3 S4  S5  S6  S7  S8  S7 . . .  R];
% VECTOR DE VARIABLES, SLACKS Y SURPLUS.

digits(5);

%%%%IDENTIFICACIÓN DE VARIABLES DE HOLGURA Y DE DECISIÓN DEL CORTE,
TRANSFORMACIÓN A VARIABLES DE DECISIÓN ORIGINALES%%%%%%%%%%%%%%%%%

X = vpa(X);

n0c11=expand(X*R1)-h;

vca1 = coeffs(n0c11);
vca2 = jacobian(n0c11);
vca3 = findsym(n0c11);

vc = [vca1(1),vca2];

for i=1:max(size(vca3))
      if ((vca3(i) ~= 'X') && (vca3(i) ~= ',' ) && (vca3(i) ~= ' ' ))

vacc(i) = vca3(i);
end
end

vacc1 = double(vacc);

k = 1;
kc(1) = 0;
j = 1;

for i=1:max(size(vca3))
```

```
      if  (vacc1(i) ~= 0)
   vacc2(k) = vacc1(i) - 48;
   k = k+1;
   j = j + 1;
   kc(j) = 0;
else

 kc(j) = kc(j) + 1;

end
end

k = 1;

for i=1:max(size(kc))
if  (kc(i) ~= 0)

   kc1(k) = kc(i);
   k = k+1;
end
end

k = 1;
kn(1) = 0;
j = 1;

for i=1:max(size(vca3))
if  (vacc1(i) ~= 0)

 kn(j) = kn(j) + 1;

else
k = k+1;
j = j + 1;
kn(j) = 0;

end
end
k = 1;
for i=1:max(size(kn))
if  (kn(i) ~= 0)

   kc2(k) = kn(i);
   k = k+1;
end
end
w = 1;
j = 1;
for i=1:max(size(kc2))
if (kc2(i) == 1)
datav(w) = vacc2(j)*1;
w = w+1;
j = j + 1;
elseif (kc2(i) == 2)
datav(w) = vacc2(j+1)*1 +
vacc2(j)*10;
w = w+1;
j = j + 2;

elseif (kc2(i) == 3)
datav(w) = vacc2(j+2)*1 +
vacc2(j+1)*10 + vacc2(j)*100;
w = w+1;
j = j + 3;
end
en
Nc = max(size(datav));
datavp1 = datav(1:Nc-1);
datavp2 = datav(2:Nc);
difdata = [abs(datavp1 - datavp2)-
1,70-datav(Nc)];

Nc = max(size(datav));
salf = zeros(1,71);
nsal = max(size(salf));
for i = 1:Nc+1
     if (i<Nc+1)
     salf(datav(i))=vc(i+1);
     else
     salf(nsal)=vc(1);
     end
 end

 ZZZ = max(size(salf));

 for i=1:ZZZ
     if (salf(i) == 0)
         salfm(i) = salf(i);
     else
         salfm(i) = -salf(i);
     end
 end

 for i = 1:ZZZ
        fprintf('%10.3f',salf(1,i))
       end
 fprintf('\n\n')

 for i = 1:ZZZ

fprintf('%10.3f',salfm(1,i))
        end
        fprintf('\n\n')



%%%%%%%%FIN DEL PROGRAMA%%%%%%%%%%
```

A continuación se presenta el programa del Método Simplex utilizado para resolver y obtener la tabla final de los problemas antes presentados.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PROGRAMA DEL METODO SIMPLEX
% METODO DE LAS DOS FASES
% TRABAJO DE TESIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BASADO EN EL LIBRO
% Applied Optimization with Matlab Programming
% Dr. P.Venkataraman
% second Edition,  John Wiley
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% management functions
clear  % clear all variable/information in the workspace - use CAUTION
clc    % position the cursor at the top of the screen
format compact  % avoid skipping a line
format rational % report results in rational form
warning off  % don't report any warnings like divide by zero etc.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

textstr1 = 'Data Entry - Set up Augmented matrix Ab = [A b]';
fprintf('Example 3.3 \n')
fprintf('Calculations using MATLAB\n')
fprintf('\n\n')
fprintf('%s',textstr1)
fprintf('\n')

c1 = 3; % variables de holgura, restricciones <=
c2 = 2; % variables holgura + artificiales , restricciones >=
c3 = 2; % variables artificiales, restricciones ==

%   X12P    X12N    Y12P   Y12N     X1         X2          P12         Q12   Y1       Y2
%   X1      X2      X3     X4       X5         X6          X7          X8    X9       X10

% MATRIZ DE RESTRICCCIONES
A = [0      0       0      0       1       -1       -100      100      0       0;
     0      0       0      0       0       0         100     -100     -1       1;
     0      0       0      0       0       0         100      100      1      -1;
     0      0       0      0       1       -1        100      100      0       0;
     -1     1       0      0       1       -1        0         0       0       0;
     0      0       -1     1       0       0         0         0       1      -1];

b = [75;80;180;25;0.3333;0;0;0;0]; %%%%% VECTOR DE LADOS DERECHOS DE LAS  RESTRICCIONES

c = [10,10,10,10,0,0,0,0,0,0];     %%%%% VECTOR DE LA FUNCION OBJETIVO

R = size(A);

Rm = [R(1),R(2) + c1 + 2*c2 + c3];

A1 = [A((1:c1),(1:R(2))),eye(c1),zeros(c1,Rm(2)-R(2)-c1),b(1:c1)];

A2 = [A((c1+1:c1+c2),(1:R(2))),zeros(c2,c1),-eye(c2),eye(c2),zeros(c2,Rm(2)-R(2)-c1-
2*c2),b(c1+1:c1+c2)];
```

```
A3 = [A((c1+c2+1:R(1)),(1:R(2))),zeros(c3,c1+2*c2),eye(c3),b(c1+c2+1:R(1))];

fa = [zeros(1,Rm(2)-c2-c3),ones(1,c2+c3),b(c1+c2+c3+2)];      % FUNCION ARTIFICIAL

fo = [c,zeros(1,Rm(2)- R(2)),b(c1+c2+c3+1)];% FUNCION OBJETIVO(+VARIABLES DE HOLGURA)

Ab = [A1;A2;A3;fo;fa];

Aart = [A2;A3];
dda=size(Aart);
if (dda(1)==1)
    Acum = Aart;
else
    Acum = cumsum(Aart);
end

    Aless = Acum(c2+c3,:);


% OBTENCION DEL NUMERO DE VARIABLES Y RESTRICCIONES A PARTIR DE LA MATRIZ AB
[m1,n1] = size(Ab);
m = m1-2;    n = n1 - 1;
fprintf('Number of variables [n]: %3i\nNumber of constraints [m]: %3i\n\n',n,m)

nmax = 121;

%%% The columns are problem dependent.  Create column symbols
%%% Done only once
coltxt = [];
for i = 1:n1
    stri = num2str(i);
    catstr = strcat('        x',stri);
    if (i == n1)
        coltxt = [coltxt,'        b'];
    else
        coltxt = [coltxt,catstr];
    end
end
%%%   columns symbol created

%%% IMPRESION DE LA TABLA DEL METODO SIMPLEX - INCLUYENDO SIMBOLOS
textstr1 = strcat('Starting Table');
fprintf('------------------\n')
fprintf('%s',textstr1)
fprintf('\n------------------')
fprintf('\n\n')
disp(coltxt)

for i = 1:m1
    fprintf('%10.3f',Ab(i,:))
    fprintf('\n')
end
fprintf('\n\n')

%-----------------------------------------------------
%%% Process the last column to create a canonical form
%-----------------------------------------------------
Ab(m1,:) = Ab(m1,:) - (Aless);%ELIMINAR LA VARIABLE ARTIFICIAL
%textstr1 = strcat('Simplex Table-',num2str(tablenumber));
```

```
%%% IMPRESION DE LA INICIAL DEL METODO SIMPLEX
tablenumber = 1;

% fprintf('------------------\n')
% fprintf('%s',textstr1)
% fprintf('\n------------------')
% fprintf('\n\n')
% %coltxt
%
% iprintsol = 0;  % Not ready for printing

  while ((Ab(m1,n+1) <= 0.001) || (Ab(m1,n+1) < -0.00001))
  %while ((Ab(m1,n+1) >= 0.001) || (Ab(m1,n+1) < -0.00001))

 % for ii = 1:nmax

    %%% Print the table - including column symbols
    textstr1 = strcat('Phase I - Simplex Table-',num2str(tablenumber));
    fprintf('------------------\n')
    fprintf('%s',textstr1)
    fprintf('\n------------------')
    fprintf('\n\n')
    disp(coltxt)

    for i = 1:m1
        fprintf('%10.3f',Ab(i,:))
        fprintf('\n')
    end
    fprintf('\n\n')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%   DECIDIR SI EL PROCESO DEBE CONTINUAR
    [minv ebv] = min(Ab(m1,1:n));
    ebvint = uint8(ebv);

    if (minv < 0.0)
        iprintsol = 0;
        fprintf('-----------------------------------\n')
        fprintf('Another Iteration is Necessary !!!!\n')
        fprintf('-----------------------------------\n')
    else
        iprintsol = 1; % Print values and stop
        fprintf('-----------------------------------\n')
        fprintf('Further Iteration is Not Possible\n')
        fprintf('Check for Solution\n')
        fprintf('-----------------------------------\n')
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if (iprintsol == 0)
        fprintf('\nEntering basic Variable (EBV) = %s\n',strcat('x',num2str(ebv)))
        % IDENTIFICACION DE LA VARIABLE QUE SALE DE LA BASE

Abp = zeros(m1);

%%%%DENOMINADORES

for j=1:m1
    if ((Ab(j,ebvint)<0) && (Ab(j,n1) >= 0))
        Abp(j,ebvint) = 0;
```

```
    elseif ((Ab(j,ebvint)<0) && (Ab(j,n1) < 0))
        Abp(j,ebvint) = Ab(j,ebvint);
    else
        Abp(j,ebvint) = Ab(j,ebvint);
    end
end

dummyvec(:,1) = Ab(:,n1)./Abp(:,ebvint);

No = max(size(dummyvec));

for i=1:No

if ((dummyvec(i) < 0) && (dummyvec(i) > -0.001))

    dummyvec(i) = 0;
end
end

        % IDENTIFICAR LA VARIABLE QUE ENTRA
        minval = +inf;   pivot = 1;
        for j = 1:m
            if (dummyvec(j,1) < minval) && (dummyvec(j,1) >= 0)
                minval = dummyvec(j,1);
                pivotrow = j;
            end
        end

        fprintf('Pivot Row: %3i\n',pivotrow)

        % IDENTIFICACION DE LA VARIABLE QUE SALE DE LA BASE
        lbv = 1;  % assume LBV is the first row

        % CREACION DE UN VECTOR PARA LA VARIABLE QUE SALE DE LA BASE
        unitvec = zeros(m1,1);
        unitvec(pivotrow,1) = 1;
        %unitvec

        %%% check the columns and capture LBV
        for i = 1:n
            if (Ab(:,i) == unitvec)
                lbv = i;
            end
        end
        %lbv

        fprintf('Leaving Basic Variable (LBV) = %s\n',strcat('x',num2str(lbv)))

        Ab(pivotrow,:)=Ab(pivotrow,:)/Ab(pivotrow,ebv);

        for j = 1:m1
            if(j ~= pivotrow)
                Ab(j,:) = Ab(j,:) - Ab(j,ebv)*Ab(pivotrow,:);
            end
        end
        %fobj = fobj - fobj*Ab(m1,ebv)*Ab(pivotrow,n1)
        tablenumber = tablenumber + 1;

    end
```

```
    if (iprintsol == 1)
        %%% RECONOCIENDO LA SOLUCION
        isol = [];
        sol = [];
        if (Ab([1:m],n1) > 0)
            % CREACION DE UN VECTOR PARA RECONOCER LAS COLUMNAS SOLUCION
            for jj = 1:m
                unitvec = zeros(m1,1);
                unitvec(jj,1) = 1;
                for i = 1:n
                    % IDENTIFICACION DE LA VARIABLE Y LA SOLUCION
                    if (Ab(:,i) == unitvec)
                          isol = [isol i];
                          sol = [sol Ab(jj,n1)];
                    end
                end
            end
            for ip = 1:m
                fprintf('\n x(%2i) = %7.3f',isol(ip),sol(ip))
            end
            fprintf('\n')
            fprintf('\nOptimal value : %10.3f',-Ab(m1,n1))
            fprintf('\n\n')
            return

        else
            fprintf('The right hand side is not all zero\n')
            fprintf('Solution may have to be interpreted\n')
            return
        end

    end
end

    %%% IMPRESION DE LA TABLA DEL METODO SIMPLEX
    textstr1 = strcat('Phase I - Simplex Table-',num2str(tablenumber));
    fprintf('-------------------\n')
    fprintf('%s',textstr1)
    fprintf('\n-------------------')
    fprintf('\n\n')
    disp(coltxt)

        for i = 1:m1
        fprintf('%10.3f',Ab(i,:))
        fprintf('\n')
        end
        fprintf('\n\n')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%FASE
2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%TOMAR LA TABLA DE LA FASE 1
%INICIALIZAR LOS VECTORES ARTIFICIALES
 Ab = [Ab((1:R(1)+1),(1:R(2)+c1+c2)),Ab((1:R(1)+1),(Rm(2)+1))];

 clear dummyvec

 clear ebvint

 % Obtain the number of variables and constraints from the Ab data
```

```
[m1,n1] = size(Ab);
m = m1-1;    n = n1 - 1;
fprintf('Number of variables [n]: %3i\nNumber of constraints [m]: %3i\n\n',n,m)

nmax = 300;
%%% IMPRESION DE LA TABLA INICIAL DEL SIMPLEX
tablenumber = 1;

%%% The columns are problem dependent.  Create column symbols
%%% Done only once
coltxt = [];
for i = 1:n1
    stri = num2str(i);
    catstr = strcat('        x',stri);
    if (i == n1)
        coltxt = [coltxt,'        b'];
    else
        coltxt = [coltxt,catstr];
    end
end
%%%   columns symbol created

%coltxt

iprintsol = 0;  % Not ready for printing
for ii = 1:nmax

    %%% IMPRESION DE LA TABLA DEL SIMPLEX EN CADA ITERACION
    textstr1 = strcat('Phase II:Simplex Table-',num2str(tablenumber));
    fprintf('------------------\n')
    fprintf('%s',textstr1)
    fprintf('\n------------------')
    fprintf('\n\n')
    disp(coltxt)

    for i = 1:m1
        fprintf('%10.3f',Ab(i,:))
        fprintf('\n')
    end
    fprintf('\n\n')

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%   DECISION SOBRE CONTINUAR O NO LA ITERACION
    [minv ebv] = min(Ab(m1,1:n));
    ebvint = uint8(ebv);


    if ((minv < 0.0) && (minv < -0.0001))
        iprintsol = 0;
        fprintf('-----------------------------------\n')
        fprintf('Another Iteration is Necessary !!!!\n')
        fprintf('-----------------------------------\n')
    else
        iprintsol = 1; % ARROJAR VALORES Y PARAR
        fprintf('-----------------------------------\n')
        fprintf('Further Iteration is Not Possible\n')
        fprintf('Check for Solution\n')
        fprintf('-----------------------------------\n')
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    if (iprintsol == 0)
        fprintf('\nEntering basic Variable (EBV) = %s\n',strcat('x',num2str(ebv)))
        %  IDENTIFICACION DE LA VARIABLE QUE SALE DE LA BASE



Abp = zeros(m1);
Abn = zeros(m1);

%%%%DENOMINADORES

for j=1:m1
    if ((Ab(j,ebvint)<0) && (Ab(j,n1) >= 0))
        Abp(j,ebvint) = 0;

    elseif ((Ab(j,ebvint)<0) && (Ab(j,n1) < 0))
        Abp(j,ebvint) = Ab(j,ebvint);
    else
        Abp(j,ebvint) = Ab(j,ebvint);
    end
end

dummyvec(:,1) = Ab(:,n1)./Abp(:,ebvint);

No = max(size(dummyvec));

for i=1:No

if ((dummyvec(i) < 0) && (dummyvec(i) > -0.0001))

    dummyvec(i) = 0;

end
end

        % IDENTIFICAR LA VARIABLE QUE ENTRA
        minval = +inf;   pivot = 1;
        for j = 1:m
            if (dummyvec(j,1) < minval) && (dummyvec(j,1) >= 0)
                minval = dummyvec(j,1);
                pivotrow = j;
            end
        end

        fprintf('Pivot Row: %3i\n',pivotrow)

        % IDENTIFICACION DE LA VARIABLE QUE SALE DE LA BASE
        lbv = 1;  % assume LBV is the first row

        % CREAR UN VECTOR PARA LA VARIABLE QUE SALE DE LA BASE
        unitvec = zeros(m1,1);
        unitvec(pivotrow,1) = 1;
        %unitvec

        %%% check the columns and capture LBV
        for i = 1:n
            if (Ab(:,i) == unitvec)
                lbv = i;
            end
        end
```

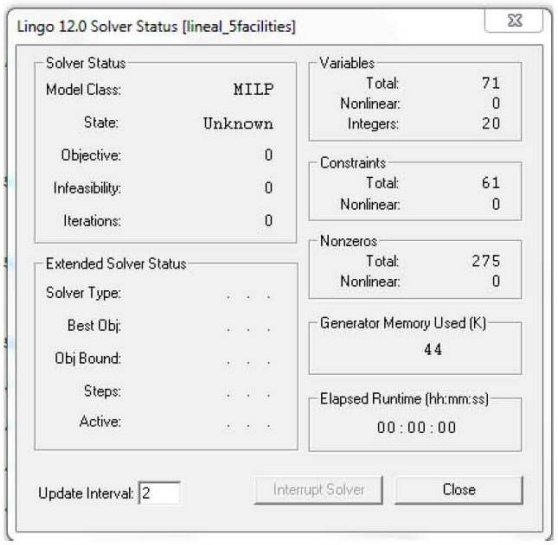
```
        %lbv

        fprintf('Leaving Basic Variable (LBV) = %s\n',strcat('x',num2str(lbv)))

        Ab(pivotrow,:)=Ab(pivotrow,:)/Ab(pivotrow,ebv);

        for j = 1:m1
            if(j ~= pivotrow)
                Ab(j,:) = Ab(j,:) - Ab(j,ebv)*Ab(pivotrow,:);
            end
        end
        %fobj = fobj - fobj*Ab(m1,ebv)*Ab(pivotrow,n1)
        tablenumber = tablenumber + 1;

    end
    if (iprintsol == 1)
        %%% RECONOCIENDO LA SOLUCION
        isol = [];
        sol = [];
        if (Ab([1:m],n1) > -0.00001)
            % CREACION DE UN VECTOR PARA RECONOCER LAS COLUMNAS SOLUCION
            for jj = 1:m
                unitvec = zeros(m1,1);
                unitvec(jj,1) = 1;
                for i = 1:n
                    % IDENTIFICACION DE LA VARIABLE Y LA SOLUCION
                    if (Ab(:,i) == unitvec)
                          isol = [isol i];
                          sol = [sol Ab(jj,n1)];
                    end
                end
            end
            for ip = 1:m
                fprintf('\n x(%2i) = %7.3f',isol(ip),sol(ip))
            end
            fprintf('\n')
            fprintf('\nOptimal value : %10.3f',-Ab(m1,n1))
            fprintf('\n\n')
            return

        else
            fprintf('The right hand side is not all zero\n')
            fprintf('Solution may have to be interpreted\n')
            return
        end

    end
end
  %%% IMPRESION DE LA TABLA FINAL DE LA FASE II DEL SIMPLEX
    textstr1 = strcat('Phase II - Simplex Table-',num2str(tablenumber));
    fprintf('------------------\n')
    fprintf('%s',textstr1)
    fprintf('\n------------------')
    fprintf('\n\n')
    disp(coltxt)

        for i = 1:m1
        fprintf('%10.3f',Ab(i,:))
        fprintf('\n')
        end
        fprintf('\n\n')
```

Ejemplo del programa en Lingo 12 para el problema que involucra 5 instalaciones.

```
MODEL:
 [_1] MIN= 10 * X12P + 10 * X12N + 10 * Y12P + 10 * Y12N + 15 * X13P +
15 * X13N +
  15 * Y13P + 15 * Y13N + 20 * X14P + 20 * X14N + 20 * Y14P + 20 *
Y14N + 30 * X23P
  + 30 * X23N + 30 * Y23P + 30 * Y23N + 35 * X24P + 35 * X24N + 35 *
Y24P + 35 *
  Y24N + 10 * X25P + 10 * X25N + 10 * Y25P + 10 * Y25N + 10 * X34P +
10 * X34N + 10
  * Y34P + 10 * Y34N + 20 * X35P + 20 * X35N + 20 * Y35P + 20 * Y35N +
15 * X45P +
  15 * X45N + 15 * Y45P + 15 * Y45N;
 [_2] X1 - X2 + 100 * P12 + 100 * Q12 >= 25;
 [_3] - X1 + X2 + 100 * P12 - 100 * Q12 >= - 75;
 [_4] X1 - X3 + 100 * P13 + 100 * Q13 >= 30;
 [_5] - X1 + X3 + 100 * P13 - 100 * Q13 >= - 70;
 [_6] X1 - X4 + 100 * P14 + 100 * Q14 >= 27.5;
 [_7] - X1 + X4 + 100 * P14 - 100 * Q14 >= - 72.5;
 [_8] X1 - X5 + 100 * P15 + 100 * Q15 >= 30;
 [_9] - X1 + X5 + 100 * P15 - 100 * Q15 >= - 70;
 [_10] X2 - X3 + 100 * P23 + 100 * Q23 >= 30;
 [_11] - X2 + X3 + 100 * P23 - 100 * Q23 >= - 70;
 [_12] X2 - X4 + 100 * P24 + 100 * Q24 >= 27.5;
 [_13] - X2 + X4 + 100 * P24 - 100 * Q24 >= - 72.5;
 [_14] X2 - X5 + 100 * P25 + 100 * Q25 >= 30;
 [_15] - X2 + X5 + 100 * P25 - 100 * Q25 >= - 70;
 [_16] X3 - X4 + 100 * P34 + 100 * Q34 >= 32.5;
 [_17] - X3 + X4 + 100 * P34 - 100 * Q34 >= - 67.5;
 [_18] X3 - X5 + 100 * P35 + 100 * Q35 >= 35;
 [_19] - X3 + X5 + 100 * P35 - 100 * Q35 >= - 65;
 [_20] X4 - X5 + 100 * P45 + 100 * Q45 >= 32.5;
 [_21] - X4 + X5 + 100 * P45 - 100 * Q45 >= - 67.5;
 [_22] - 100 * P12 + 100 * Q12 + Y1 - Y2 >= - 80;
 [_23] - 100 * P12 - 100 * Q12 - Y1 + Y2 >= - 180;
 [_24] - 100 * P13 + 100 * Q13 + Y1 - Y3 >= - 75;
 [_25] - 100 * P13 - 100 * Q13 - Y1 + Y3 >= - 175;
 [_26] - 100 * P14 + 100 * Q14 + Y1 - Y4 >= - 80;
 [_27] - 100 * P14 - 100 * Q14 - Y1 + Y4 >= - 180;
 [_28] - 100 * P15 + 100 * Q15 + Y1 - Y5 >= - 80;
 [_29] - 100 * P15 - 100 * Q15 - Y1 + Y5 >= - 180;
 [_30] - 100 * P23 + 100 * Q23 + Y2 - Y3 >= - 75;
 [_31] - 100 * P23 - 100 * Q23 - Y2 + Y3 >= - 175;
 [_32] - 100 * P24 + 100 * Q24 + Y2 - Y4 >= - 80;
 [_33] - 100 * P24 - 100 * Q24 - Y2 + Y4 >= - 180;
 [_34] - 100 * P25 + 100 * Q25 + Y2 - Y5 >= - 80;
 [_35] - 100 * P25 - 100 * Q25 - Y2 + Y5 >= - 180;
 [_36] - 100 * P34 + 100 * Q34 + Y3 - Y4 >= - 75;
 [_37] - 100 * P34 - 100 * Q34 - Y3 + Y4 >= - 175;
 [_38] - 100 * P35 + 100 * Q35 + Y3 - Y5 >= - 75;
 [_39] - 100 * P35 - 100 * Q35 - Y3 + Y5 >= - 175;
 [_40] - 100 * P45 + 100 * Q45 + Y4 - Y5 >= - 80;
 [_41] - 100 * P45 - 100 * Q45 - Y4 + Y5 >= - 180;
 [_42] - X12P - X12N + X1 - X2 = 0;
 [_43] - X13P - X13N + X1 - X3 = 0;
 [_44] - X14P - X14N + X1 - X4 = 0;
 [_45] X1 - X5 - X15P - X15N = 0;
 [_46] - X23P - X23N + X2 - X3 = 0;
 [_47] - X24P - X24N + X2 - X4 = 0;
 [_48] - X25P - X25N + X2 - X5 = 0;
 [_49] - X34P - X34N + X3 - X4 = 0;
 [_50] - X35P - X35N + X3 - X5 = 0;
```

```
  [_51] X4 - X5 - X45PX45N = 0;
  [_52] - Y12P - Y12N + Y1 - Y2 = 0;
  [_53] - Y13P - Y13N + Y1 - Y3 = 0;
  [_54] - Y14P - Y14N + Y1 - Y4 = 0;
  [_55] Y1 - Y5 - Y15P - Y15N = 0;
  [_56] - Y23P - Y23N + Y2 - Y3 = 0;
  [_57] - Y24P - Y24N + Y2 - Y4 = 0;
  [_58] - Y25P - Y25N + Y2 - Y5 = 0;
  [_59] - Y34P - Y34N + Y3 - Y4 = 0;
  [_60] - Y35P - Y35N + Y3 - Y5 = 0;
  [_61] - Y45P - Y45N + Y4 - Y5 = 0;
END
```

Resultados obtenidos con LINGO 12.0




```
Global optimal solution found.
  Objective value:                              5525.000
  Objective bound:                              5525.000
  Infeasibilities:                              0.000000
  Extended solver steps:                           26845
  Total solver iterations:                        191126

  Model Class:                                      MILP

  Total variables:                    70
  Nonlinear variables:                 0
  Integer variables:                  20

  Total constraints:                  61
  Nonlinear constraints:               0

  Total nonzeros:                    276
  Nonlinear nonzeros:                  0
```