UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE CIENCIAS

"Asignación de horarios a enfermeras con preferencias".

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
A C T U A R I A
P R E S E N T A :

BRENDA BERENICE CARRASCO ENRÍQUEZ

DIRECTOR: M. en C. JESÚS AGUSTÍN CANO GARCÉS

2007

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Hoja de datos del jurado

1. Datos del alumno
   Autor
   Carrasco
   Enríquez
   Brenda Berenice
   55 28 56 07
   Universidad Nacional Autónoma de México
   Facultad de Ciencias
   Actuaría
   098017607

2. Datos del asesor
   M. en C.
   Cano
   Garcés
   Jesús Agustín

3. Datos del sinodal 1
   Mat.
   Adrián
   Girard
   Islas

4. Datos del sinodal 2
   Act.
   Amalia
   Maldonado
   Rosas

5. Datos del sinodal 3
   Act.
   Germán
   Valle
   Trujillo

6. Datos del sinodal 4
   Act.
   Claudia Orquídea
   López
   Soto

7. Datos de la tesis.
   Asignación de horarios a enfermeras con preferencias
   107 p.
   2007

Dios, por ser la luz que ilumina mi camino, te dedico mi tesis.

Papás, gracias por la familia que formaron para mi y mis hermanas. Por el amor que se tienen y que nos han dado en cada momento.

Papá, gracias por tus consejos, tu fuerza y tu apoyo cuando más los necesito.

Mamá, gracias por tu dedicación, por tu infinito amor y por la tranquilidad que siento cuando estoy contigo.

Arlen, gracias por crecer a mi lado, por las pláticas y los aprendizajes que hemos vivido juntas.

Xanat, gracias por tu alegría, tu sonrisa, tu frescura y tus enseñanzas.

Prof. Cano, gracias por su tiempo y conocimientos.

Asignación de horarios a enfermeras con preferencias

Índice

Introducción

Desde hace tiempo, al ver la efectividad que los métodos de optimización han tenido sobre la distribución y organización de los recursos físicos de los procesos productivos, se ha intentado hacer lo mismo con los recursos humanos de los que disponen las empresas.

En toda empresa o institución que presta servicios, ya sea pública o privada, es muy importante poder hacer "el mejor" uso posible de los recursos disponibles en ella, promoviendo servicios de alta calidad al menor costo. El caso de las instituciones médicas no es la excepción, y el personal de enfermería es uno de los factores en los que se destinan más recursos, por lo que es necesario poder optimizar el número de enfermeras requeridas o la forma en la que se organizan sus horarios, ya que esto implica un costo. Dicho costo puede ser de carácter económico o de carácter cualitativo, es decir, que represente la aversión por parte de las enfermeras a cierto horario, lo cual repercute directamente en el ánimo del personal.

El problema de organización de horarios, también conocido como "The Scheduling Problem", "Rostering" o "Employee Timetabling Problem" atiende a esta problemática y consiste por un lado en la construcción de patrones de trabajo para empleados y por otro en la asignación de estos, durante un periodo fijo de tiempo.

El problema de organización de horarios ha sido un problema ampliamente abordado, sobre todo porque existen variaciones de éste, originadas por una gran diversidad de factores particulares que modifican el problema original, como lo son los arreglos contractuales de cada institución, la legislación vigente en el país o lugar en cuestión, así como los requerimientos y las restricciones de las enfermeras, que impiden que una sola metodología de resolución sea capaz de contemplar todos los casos de una manera eficiente.

A diferencia de otros ambientes laborales en los que horarios cíclicos satisfacen todas las necesidades y requerimientos, en los hospitales surgen dos problemáticas: por un lado se necesita de mayor flexibilidad en los horarios ya que estos tienen que considerar los distintos tipos de atención que requieren los pacientes y por consiguiente los distintos tipos de enfermeras que se necesitan, además de los tipos de turnos y las regulaciones laborales particulares al cuidado de la salud. Por otro lado los hospitales en el mundo afrontan un problema que es el de la falta de personal de enfermería que pueda satisfacer las demandas de cuidado que el hospital requiere. Lo que ha propiciado por consiguiente el encarecimiento de éste servicio.

Por un lado los hospitales buscan proveer un mínimo de atención en términos de personal de enfermería y por otro lado las enfermeras quieren "buenos" horarios. El conflicto surge cuando ésta cobertura de personal sólo se satisface negando a las enfermeras ciertos días libres requeridos por ellas o empleando patrones de trabajo considerados indeseables Warner(1976).

Como una forma de poder contrarrestar la escasez de personal y los conflictos antes mencionados, muchos hospitales han optado por políticas en la generación de horarios que tomen en cuenta preferencias y requerimientos del personal de enfermería, obviamente que a un costo. Lo que propicia mejores condiciones de trabajo para las enfermeras y en consecuencia una mejor retención de personal.

Para intentar resolver un problema como éste que es de carácter real, no es necesario trabajar directamente con él. El punto es por un lado encontrar un modelo en el que se puedan representar todas las características fundamentales del problema para así acercarnos al panorama que prevalece en la realidad. Por otro, hallar una metodología de solución que optimice ésta labor.

Actualmente existe una gran variedad de metodologías y modelos que atienden a las variaciones concernientes a las particularidades de cada hospital y que resuelven este tipo de problemas. Estas metodologías abarcan desde la programación matemática hasta métodos con meta-heurísticas y técnicas de satisfacción de restricciones.
El problema de organización de horarios para enfermeras consiste en la asignación diaria de turnos a enfermeras, en un periodo de tiempo, tomando en cuenta la demanda de enfermeras por tipo de categoría, requerimientos de días libres, fines de semana y patrones de trabajo. Es decir, lo que se busca es construir un horario que satisfaga lo más posible todos los requerimientos planteados y que minimice el costo de la asignación tomando en cuenta los recursos disponibles.

Este trabajo busca presentar una metodología propuesta recientemente para resolver el problema de la asignación de horarios a enfermeras con preferencias. Esta metodología emplea técnicas de generación de columnas que combinan el uso de programación entera con heurísticas. Además, propone una forma de cuantificación de violaciones a preferencias que no implica mucho esfuerzo por parte del usuario, y que en consecuencia facilita el cálculo de los coeficientes de costo de la función objetivo del modelo.

Para alcanzar este objetivo se han contemplado 4 capítulos. En el capítulo 1 se describe lo que es el problema de asignación y algunas aplicaciones relacionadas al proceso de la organización de horarios.

En el capítulo 2 se presentan algunas metodologías utilizadas en la solución de problemas de asignación enfocadas al proceso de organización de horarios, dándose una importancia particular a la metodología de generación de columnas. El objetivo es hacer notar la diferencia de ésta con la metodología heurística de generación de columnas presentada en éste trabajo.

En el capítulo 3 se aborda el problema de la organización de horarios y se presenta el modelo utilizado para resolver el problema de asignación contemplando las preferencias de las enfermeras.

En el capítulo 4 se describe la metodología de solución y se presentan algunos ejemplos de la aplicación de ésta.

Capítulo 1

Problemas de Asignación

El problema de asignación es un problema muy abordado debido a la importancia que tiene para las empresas, ya que de manera general, el problema trata de asignar un conjunto de recursos limitados a un conjunto de actividades de la mejor manera posible.
Una buena asignación puede representar un ahorro significativo en los gastos del sistema en el que se esté trabajando. De aquí que el campo de aplicación de éste problema sea tan extenso, particularmente en la industria.

El problema de asignación es un caso especial del problema de transporte, el cual es un caso particular de los problemas de flujo de costo mínimo.

1.1 Problema de transporte

El problema de transporte consiste en enviar mercancía de *m* orígenes a *n* destinos. Cada origen *i* dispone de a_i unidades de dicha mercancía para enviar, mientras que cada destino *j* demanda b_j unidades de dicha mercancía. El costo de enviar cada unidad de mercancía del origen *i* al destino *j* se denota por c_{ij}. El objetivo es minimizar el costo total de transportación de la mercancía hallando la mejor combinación posible de envíos de orígenes a destinos. Sea define x_{ij} como número de unidades de mercancía que se deben enviar desde el origen *i* al destino *j*.

El modelo matemático para el problema de transporte es el mostrado a continuación.

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

$$\text{s.a.} \quad \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1, \ldots, m.$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad j = 1, \ldots, n.$$

$$x_{ij} \geq 0 \text{ para toda } i \text{ y para toda } j.$$

Para que el problema tenga solución basta con que la demanda y la disponibilidad totales de mercancía estén balanceadas, es decir, $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. Cuando no es así, y la disponibilidad excede a la demanda basta con agregar un destino ficticio con costo de envío a él igual a 0, con lo que se obtiene el balance deseado. Si la demanda excede la disponibilidad no existe solución al problema.

1. 2 Definición del problema de asignación

El término de asignación proviene de la aplicación clásica de asignar un conjunto de trabajadores a un conjunto de trabajos. Como ya se mencionó este es un caso particular del problema de transporte en donde se consideran, por un lado, igual número de orígenes y destinos y por otro demanda y disponibilidad de mercancía iguales a una unidad en todos los casos. Es decir, se tiene un conjunto de *m* trabajadores que se deben asignar de la manera más eficiente posible a otro conjunto de *m* trabajos.

*S*ea x_{ij} una variable binaria que toma el valor de 1 si el trabajador *i* se asigna al trabajo *j* y 0 si no es asignado. Mientras que c_{ij} representa el costo de dicha asignación.

Cada trabajo debe asignarse a uno y sólo un trabajador. Cada trabajador realizará uno y sólo un trabajo. El problema es por tanto, decidir el modo en el que se deben realizar todas las asignaciones para minimizar los costos totales.

El modelo matemático para el problema de asignación es el mostrado a continuación.

$$Min \qquad \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\text{s.a.} \quad \sum_{i=1}^{m} x_{ij} = 1 \quad j = 1, \ldots, m.$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \ldots, m.$$

$$x_{ij} = 0 \ \acute{o} \ 1 \quad i = 1, \ldots, m, \ j = 1, \ldots, m$$

En forma matricial se tiene

$$Min\ cx$$

$$s.a.\ Ax = b$$

$$x \text{ binaria}$$

En donde A tiene la siguiente estructura especial ilustrada en la figura 1.2.1:

Fig. 1.2.1 Estructura de la matriz A del problema de asignación

Al problema de asignación también se le puede asociar un modelo de maximización, en el cual los coeficientes c_{ij} representan el beneficio obtenido al asignar al trabajador *i* al trabajo *j*, dicho beneficio puede estar en función de las capacidades particulares de cada trabajador u otras características de interés para la empresa que se reflejan en productividad. Así, en lugar de tener una matriz de costos se tendrá una matriz de utilidades. El objetivo será hallar la mejor asignación posible que maximice los beneficios.

Otra variante de éste problema es aquella en donde la solución nos dirá cómo asignar a los trabajadores de tal forma que se minimice el número de horas trabajadas. En donde los coeficientes c_{ij} denotaran el tiempo que tarda el trabajador *i* al realizar una cierta tarea *j*.

1.3 Unimodularidad total de la matriz de asignación

Una propiedad importante de la matriz de asignación es la propiedad de unimodularidad total. Es decir, una matriz es totalmente unimodular si el determinante de cualquier submatriz cuadrada es igual a -1, 0 ó 1. Esto permite en consecuencia reemplazar la restricción $x_{ij} = 0\ ó\ 1$del problema de asignación por la restricción $x_{ij} \geq 0$ y aún así la solución básica óptima al problema de asignación será entera. Es decir x_{ij} seguirá tomando el valor de 0 ó 1 para toda *i* y toda *j* Bazaraa(1990).

1.4 Algoritmo Húngaro de Kuhn

El Algoritmo Húngaro de Kuhn (1955) es uno de los más utilizados para resolver problemas de asignación. El algoritmo consiste en los siguientes pasos:

1. Crear una matriz denominada reducida a partir de la matriz de costos del problema de asignación. Ésta se obtiene primero de restar el mínimo valor de cada renglón y luego restar el mínimo valor de cada columna de cada elemento de ésta.
2. Se dibuja el mínimo número de líneas posibles a través de los renglones y columnas de la matriz reducida de tal forma que se cubran todos los ceros de la matriz. Si el número de líneas es igual a *m* (número de asignaciones a realizar) la asignación actual es óptima, de no ser así se pasa al paso 3.
3. Se selecciona el mínimo elemento de la matriz sin cubrir. Se resta éste valor de cada uno de los elementos sin cubrir de la matriz y se suma a cada uno de los elementos que fueron cubiertos por dos líneas simultáneamente. Se regresa al paso 2.

Debido a que el problema de asignación es una clase especial de problemas de flujo, muchos algoritmos para resolver éste problema pueden ser vistos como adaptaciones de algoritmos del problema de flujo de mínimo costo. Algunos de estos algoritmos son el algoritmo de relajación (relaxation algorithm) y el algoritmo de camino corto sucesivo (successive shortest path algorithm) Ahuja(1993).

1.5 Problema de asignación generalizada

El problema consiste en minimizar el costo de la asignación de *n* tareas a *m* recursos (empleados, máquinas, etc.), con capacidad limitada, de tal forma que cada trabajo sea asignado solamente a un recurso sujeto a las restricciones de capacidad. Con esto lo que se busca aumentar es la productividad de los recursos existentes y al mismo tiempo mejorar la calidad del servicio que se esté prestando si ese es el caso Ramalhinho Dias (2004).

La importancia de éste modelo radica en que aparece como subestructura en muchos modelos desarrollados para resolver problemas reales en áreas industriales o de transporte Ahuja(1993).

El modelo matemático para el problema de asignación generalizada es el mostrado a continuación.

$$Min \qquad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij}$$

$$\text{s.a.} \qquad \sum_{i=1}^{m} x_{ij} = 1 \quad j = 1, \ldots, n.$$

$$\sum_{j=1}^{n} w_{ij} x_{ij} = a_i \quad i = 1, \ldots, m.$$

$$x_{ij} = 0 \text{ ó } 1 \quad i = 1, \ldots, m, \; j = 1, \ldots, n$$

Donde c_{ij} es el costo de asignar la tarea i al recurso j, w_{ij} es la demanda sobre la capacidad del recurso i por la tarea j si ésta es asignada al recurso i , por ejemplo, si hablamos de tiempo y de trabajadores, w_{ij} sería el tiempo que necesitaría el trabajador i para realizar la tarea j en caso de ser asignado a ésta tarea y a_i es la capacidad del recurso i, que en el caso del ejemplo sería el tiempo total del que dispone el trabajador i.

Las primeras restricciones garantizan que cada tarea sea asignada a un solo recurso que requiera una cierta cantidad de la capacidad de éste, mientras que las otras restricciones están relacionadas con la capacidad de los recursos ya que por ejemplo en las industrias existen diversos materiales y equipamientos que tienen una elevada utilización y un alto costo por lo que son limitados.

1.6 Otros problemas de asignación

1.6.1 Problema de asignación de cuello de botella “The Bottleneck Assignment Problem”

Este problema es una variante importante del problema clásico de asignación antes mencionado. Una posible interpretación del problema es la dada por Garfinkel(1970) en la cual se tiene una línea de producción con m trabajos, en donde cada uno requiere de un hombre que lo lleve a cabo. Si se define c_{ij} como el tiempo que tarda el hombre i en realizar la tarea j. El objetivo es minimizar el tiempo de terminación total de los trabajos asumiendo que todos se inician al mismo tiempo.

Para una asignación dada, el tiempo de terminación de la línea de producción está dado por el máximo c_{ij} en la asignación, el cual se busca minimizar sobre todas las asignaciones. Lo que se intenta es determinar la asignación completa para la cual la asignación más tardada sea tan pequeña como sea posible Ahuja (1993).

El modelo matemático para el problema de asignación de cuello de botella es el mostrado a continuación.

$$Min \; max_{i,j} \; c_{ij} x_{ij}$$

$$\text{s.a.} \quad \sum_{i=1}^{m} x_{ij} = 1 \quad j = 1, \ldots, m.$$

$$\sum_{j=1}^{m} x_{ij} = 1 \quad i = 1, \ldots, m.$$

$$x_{ij} = 0 \text{ ó } 1 \quad i = 1, \ldots, m, \; j = 1, \ldots, m$$

Donde x_{ij} es una variable binaria que toma el valor de 1 si al trabajador *i* se le asigna la tarea *j* y 0 si no.

1.6.2 Problema de asignación cuadrática

Este problema de asignación ha sido ampliamente estudiado en optimización combinatoria por ser un problema difícil de resolver. El problema consiste en asignar *n* facilidades a *n* localidades. El objetivo es minimizar el costo total de asignación. Este costo se relaciona directamente con el costo de asignar una cierta facilidad a una cierta localidad y el costo asociado a la cantidad de flujo entre cada par de facilidades y las distancias asociadas a las localidades asignadas Koopmans(1957).

El modelo matemático para el problema de asignación cuadrática es el mostrado a continuación.

$$Min \sum_{i,j=1}^{n} \sum_{k,h=1}^{n} d_{ih} f_{jk} x_{ij} x_{hk} + \sum_{i,j=1}^{n} c_{ij} x_{ij}$$

$$s.a. \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n.$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n.$$

$$x_{ij} = 0 \text{ ó } 1 \quad i = 1, \ldots, n, \; j = 1, \ldots, n$$

Donde d_{ih} representa la distancia entre localidades, f_{jk} el flujo entre las facilidades, c_{ij} representa el costo de asignar facilidades a localidades y x_{ij} es una variable binaria que toma el valor de 1 si se hace la asignación y 0 si no.

Como ejemplo se considerará el descrito en Panos(2004), en donde el problema consiste en edificar nuevas departamentos (facilidades) en sitios (localidades) disponibles en un campus universitario. El objetivo es asignar la construcción de los departamentos a los sitios disponibles de tal forma que se minimice el costo de la asignación y la distancia total a caminar para los estudiantes y el personal de la universidad. Se consideran *n* localidades

disponibles y *n* facilidades a edificar. Donde d_{ih} representa la distancia caminando entre las localidades *i* y *h* en donde se edificaran las facilidades, f_{jk} representa la cantidad de gente que viaja entre las facilidades *j* y *k* y c_{ij} representa el costo de construir la facilidad *i* en la localidad *j*.

1.7 Aplicaciones

Aunque el modelo de asignación clásico asigna empleados a tareas, su uso se ha ampliado a una gran diversidad de industrias, entre sus principales aplicaciones están las relacionadas a los problemas de "scheduling".

El término "scheduling" se ha asociado a la actividad de colocar entidades (personal, tareas a elaborar, vehículos, clases, exámenes) dentro de patrones que consideran tiempo y espacio, de tal forma que se satisfagan un conjunto de restricciones y se logren alcanzar ciertos objetivos definidos Wren(1996).

Dentro de éstas aplicaciones se encuentran las asignaciones de trabajadores como la asignación de tripulación a vuelos (Airline Crew Assignment), Asignación de trabajos de producción a máquinas (Machine assignment), Asignación de productos a fabricar (Production scheduling), Construcción de horarios (Timetabling Problems), y la organización de personal como la organización de enfermeras (Nurse Scheduling), etc.

1.7.1 Problema general de asignación de personal Gass(1985)

Frecuentemente existen trabajos que son idénticos y que demandan los mismos requisitos básicos para realizarse. Dichos trabajos pueden ser divididos en categorías. Se asume que existen *n* de esas categorías y que b_j denota el número de trabajos agrupados en la *j-ésima* categoría. Si distintos individuos tienen idénticas o aproximadamente las mismas cualidades para realizar un trabajo, es decir, los mismos valores de c_{ij}, estos individuos pueden ser agrupados en categorías de personal. Se asume que existen m de estas categorías y que a_i denota el número de personas en la *i-ésima* categoría de personal.

El modelo matemático que representa este problema de asignación es el mostrado a continuación.

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

$$\text{s.a.} \quad \sum_{j=1}^{n} x_{ij} = a_i \quad i = 1, \ldots, m.$$

$$\sum_{i=1}^{m} x_{ij} = b_j \quad j = 1, \ldots, n.$$

$$x_{ij} \geq 0 \text{ para toda } i \text{ y para toda } j.$$

Donde x_{ij} indica el número de personas de la categoría de personal i asignadas a trabajar en la categoría de trabajo j. Se supone que el total de trabajos disponibles es igual al número de personas a ser asignadas, es decir, $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$.

1.7.2 Problema de asignación balanceado Ahuja (1993)

Este problema también es una variante importante del problema clásico de asignación. Dadas *n* personas y *n* tareas, se define c_{ij} como la cantidad de trabajo que la persona *i* requiere para realizar la tarea *j*. El interés que se tiene es el de escoger parejas de personas y trabajos que distribuyan la carga de trabajo de forma tan equitativamente como sea posible. Otro ejemplo de este tipo de problema es considerar c_{ij} como el tiempo de vida esperado para una componente *j* producida por una compañía *i*. Se desea escoger las componentes de tal forma que el reemplazo de todas las componentes se haga al mismo tiempo. El objetivo es determinar la asignación total que minimice la diferencia entre la asignación más costosa y la menos costosa.

1.7.3 Asignación de trabajos de producción a máquinas (Machine Assignment Problem) Gass(1985)

Esta aplicación se refiere a problemas en donde se tienen que asignar *n* tareas para ser procesadas en *n* distintas máquinas. Se define c_{ij} como el costo o el tiempo de realización que cuesta o que tarda la maquina *j* en realizar la tarea *i*. El objetivo es encontrar la asignación de trabajos a máquinas de tal forma que se minimice el tiempo total de realización o el costo según sea el caso.

1.7.4 Problema del matrimonio estable (Stable Marriage Problem)

Este problema se considera una extensión del problema de asignación. En su forma clásica se tiene una comunidad que consiste de *n* hombres y de *n* mujeres. Cada mujer y cada hombre proporcionan una lista de preferencias sobre los miembros del sexo opuesto. El problema consiste en formar matrimonios estables. Un matrimonio estable es un conjunto de parejas entre estos hombres y mujeres tales que no existe un hombre que prefiera más a otra mujer que a su pareja y que ésta lo prefiera más sobre la de ella. Es decir, puede suceder que exista un hombre que prefiera más a otra mujer que a su pareja pero no habrá mujer que lo prefiera más a él que a la pareja de ella y viceversa. El problema consiste entonces en hallar un emparejamiento perfecto en el que halla sólo matrimonios estables.

Como en el problema de asignación lo que se busca es encontrar el emparejamiento (asignación) que maximice las preferencias.

$$max \sum_{h=1}^{n}\sum_{m=1}^{n} P(h,m)x_{hm} = max \sum_{h=1}^{n}\sum_{m=1}^{n} (P_h(h,m) + P_m(h,m))x_{hm}$$

Donde $P(h,m)$ representa qué tanto aceptan tanto el hombre *h* como la mujer *m* el emparejamiento y es una función de la preferencia que tiene el hombre *h* por la mujer *m* y la preferencia que tiene la mujer *m* por el hombre *h*.

1.7.5 Construcción de horarios (Timetabling Problems)

La construcción de horarios consiste en la asignación de recursos a objetos situados en el tiempo y espacio, estando sujeto a restricciones, de tal forma que se satisfaga lo más posible un conjunto de objetivos deseados. La aplicación que ha sida más estudiada es la construcción de horarios académicos (Academic Timetabling Problems) Gunawan(2006).

El problema tiene un gran número de variantes, dependiendo del país, del tipo de escuela y de las particularidades de ésta Schaerf(1996). Dentro de la construcción de horarios académicos se presentan dos tipos de problemas de asignación: los relacionados con la asignación de maestros a cursos según sus aptitudes y los problemas relacionados con la asignación de cursos a salones de clases de acuerdo a la disponibilidad de espacio o requerimientos especiales para la clase.

Para el primer problema se deben considerar las preferencias de los maestros por las materias que desean impartir, que todos los cursos sean asignados a un profesor y que no se sobrecarguen las asignaciones de cursos a algunos profesores. Para el segundo problema se considera qué el tamaño del salón cubra los requerimientos de espacio para el número de alumnos que tomarán la clase en cuestión.

Capítulo 2

Metodologías utilizadas en la solución de problemas de asignación

En la práctica, el encontrar soluciones eficientes cuando se habla de aplicaciones de problemas de asignación y de optimización combinatoria no es una tarea fácil. Las restricciones que se involucran en problemas reales al tratar de buscar que los modelos se ajusten lo mejor posible a la situación que prevalece y el tamaño de estos problemas hacen que la búsqueda de soluciones o la construcción de éstas sea una tarea difícil. Para resolver esto, diversas técnicas se han utilizado en el desarrollo de nuevas metodologías para cada uno de éstos problemas: asignación de tripulación a vuelos (Airline Crew Assignment), asignación de trabajos de producción a máquinas (Machine assignment), Construcción de horarios (Timetabling Problems) y organización de personal. En este capítulo se exponen algunas técnicas que se han utilizado en el desarrollo de nuevas metodologías para dar solución a este tipo de problemas.

2.1 Generación de columnas

Antes de comenzar a detallar dicha metodología, se abordarán algunos temas básicos concernientes al método de generación de columnas.

2.1.2 Principio de descomposición de Dantzig y Wolfe (Hadley 1962)

El principio de descomposición de Dantzig y Wolfe es una metodología utilizada para resolver problemas de programación lineal clasificados en dos tipos: los denominados de gran escala debido a su complejidad, ya sea por el número de variables o el número de restricciones involucradas y aquellos cuya estructura particular hace que la aplicación de la metodología haga mucho más eficiente la búsqueda de la solución del problema.
En los problemas de gran escala la solución puede ser encontrada de manera más rápida porque la técnica disminuye significativamente el número de operaciones a llevar a cabo, sobre todo si la matriz de coeficientes tiene una estructura particular. Este tipo de técnicas de programación lineal son también muy eficientes en el caso de problemas de programación entera, aunque la aplicación no sea de manera tan directa, y por consiguiente aplicables a la solución de problemas de organización de horarios.

El principio de descomposición se basa en la posibilidad de formar un problema equivalente al original que consta de un número menor de restricciones y de la partición del problema principal en dos partes. La primera consiste en un problema maestro y la segunda en una serie de subproblemas cuya solución individual es mucho más fácil de encontrar debido a su simplicidad o a la estructura de éstos. De manera informal se puede decir que el papel que juegan estos problemas en la descomposición es que el problema maestro genera un conjunto de coeficientes de costo para cada subproblema y éstos a su vez generan columnas candidatas a formar parte de la solución del problema maestro.

La diversidad de estas técnicas radica en la estructura particular de la matriz de coeficientes o en la metodología utilizada para aprovechar dicha estructura, pero todas ellas se basan en el principio de descomposición clásico de Dantzig y Wolfe, el cual aprovecha la estructura de la matriz de coeficientes de los problemas, la cual es angular por bloques. Es decir, una matriz de coeficientes cuya estructura es como la mostrada en la figura 1.

$$A = \begin{bmatrix} A_1 & A_2 & A_3 & \dots & A_r \\ B_1 & & & & \\ & B_2 & & & \\ & & B_3 & & \\ & & & & \\ & & & & B_r \end{bmatrix}$$

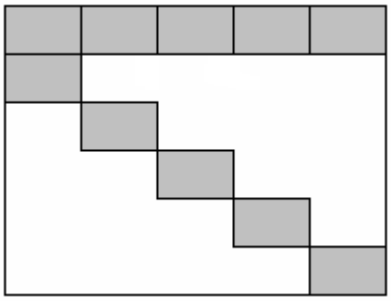


Fig. 1 Matriz angular por bloques

Las matrices de éste tipo dan lugar a sistemas de ecuaciones de la forma:

$$\begin{aligned} A_1x_1 + A_2x_2 + \dots + A_rx_r &= bo \\ B_1x_1 &= b_1 \\ B_2x_2 &= b_2 \\ &\vdots \\ B_rx_r &= b_r \end{aligned}$$

En donde, cada A_j es una matriz de m_o x n_j componentes, B_j de m_j x n_j, las dimensiones de b_o y de b_j son de m_o y m_j componentes respectivamente y las de x_j y c_j de n_j componentes. De esta forma, se tiene uno o más bloques independientes y un conjunto de ecuaciones que los liga.

Si no se tuviera el conjunto de restricciones que liga a todos los bloques, el problema se podría resolver optimizando de manera independiente r subproblemas, pero debido a que los A_j de la matriz difícilmente son todos ceros esto no es posible, pero el método de descomposición lo resuelve.

De esta forma el método de descomposición de Dantzig y Wolfe se puede aplicar a problemas cuya estructura es la siguiente:

$$\begin{aligned} Max\ z = & c_1x_1 + c_2x_2 + \dots + c_rx_r \\ & s.a. \sum_{i=1}^{r} A_ix_i = bo \\ & B_ix_i = b_i \quad i = 1,\dots, \text{r} \\ & x_i \geq 0 \end{aligned} \qquad (1)$$

Ahora bien, el método se basa en la posibilidad de utilizar el Teorema de representación y el hecho de que el conjunto de puntos $x_i \geq 0$ que satisfacen $B_i x_i = b_i$, por ser conjunto de soluciones factibles es un conjunto convexo y cerrado, que tiene un número finito de puntos extremos. De esta forma si el conjunto de soluciones factibles es acotado, cualquier punto en el conjunto puede ser representado como combinación lineal de dichos puntos, es decir, cualquier punto x_j del conjunto, puede ser escrito de la forma:

$$x_j = \sum_{i=1}^{h_j} \lambda_{ij} x_{ij}^*$$

$$\sum_{i=1}^{h_j} \lambda_{ij} = 1$$

$$\text{con } \lambda_{ij} \geq 0 \text{ e } i = 1,\ldots,h_j \text{ y } \quad j = 1,\ldots,r$$

En donde, los x_{ij}^* son los puntos extremos de los conjuntos y h_j se supone como el número de puntos extremos.

Así, bajo estos términos se puede plantear un problema equivalente a (1) de la siguiente manera:

$$Max\ z = \sum_{j=1}^{r} \sum_{i=1}^{h_j} c_j (\lambda_{ij} x_{ij}^*)$$

$$\text{s.a. } \sum_{j=1}^{r} \sum_{i=1}^{h_j} A_j (\lambda_{ij} x_{ij}^*) = bo \qquad (2)$$

$$\sum_{i=1}^{h_j} \lambda_{ij} = 1, \quad j = 1,\ldots,r.$$

$$\text{con } \lambda_{ij} \geq 0, \quad i = 1,\ldots,h_j$$

Como cualquier combinación lineal de los puntos extremos del conjunto forma parte del conjunto de soluciones factibles, x_j resulta ser una solución factible a $B_j x_j = b_j$. Al encontrar cualquier solución factible que satisfaga las ecuaciones de (2), es decir un conjunto de $\lambda_{ij} \geq 0$, se determinará un conjunto de $x_j \geq 0$ que satisface las ecuaciones de (1). Por lo que ambos problemas son equivalentes (Hadley, 1962).

En el caso de que las regiones comprendidas por los x_i que satisfacen $B_i x_i = b_i$ no sean acotadas, cualquier punto x_j del conjunto se puede escribir como una combinación lineal de los puntos extremos y de los rayos extremos de la siguiente forma:

$$x_j = \sum_{i=1}^{h_j} \lambda_{ij} x_{ij}^* + \sum_{i=1}^{r_j} \theta_{ij} w_{ij}^* \text{ , con } \lambda_{ij} \geq 0 \text{ , } \theta_{ij} \geq 0$$

$$\sum_{i=1}^{h_j} \lambda_{ij} = 1, \quad j = 1,\ldots, r.$$

De igual forma se reformula el problema original sustituyendo cada x_j por la expresión anterior. En donde, los x_{ij}^* son los puntos extremos de los conjuntos, w_{ij}^* los rayos extremos y h_j y r_j se suponen como el número de puntos y de rayos extremos respectivamente de cada conjunto.

Al llevar a cabo la transformación con los puntos extremos, se logra reducir considerablemente el tamaño del problema al disminuirse el número de restricciones comparadas con las del problema original. Aunque por otro lado el número de variables se incrementa esto no es un problema ya que como se detallará más adelante existe una forma de solucionarlo.

De esta forma se pasó de un problema con $\sum_{i=0}^{r} m_i$ restricciones a uno con (m_0+ r) restricciones.

Normalmente, para simplificar la notación, cuando el conjunto de soluciones factibles es acotado, se suele definir $d_{ij} = A_j x_{ij}^*$ y $f_{ij} = c_j x_{ij}^*$, quedando el siguiente modelo denominado problema maestro:

$$Max \ z = \sum_{j=1}^{r} \sum_{i=1}^{h_j} f_{ij} \lambda_{ij}$$

s.a.

$$\sum_{j=1}^{r} \sum_{i=1}^{h_j} d_{ij} \lambda_{ij} = bo \quad i = 1,\ldots,h_j. \ \ldots..$$

$$\sum_{i=1}^{h_j} \lambda_{ij} = 1 \qquad j = 1,\ldots,r$$

Escribiendo las restricciones del problema de manera general se tiene:

$$\sum_{j=1}^{r} \sum_{i=1}^{h_j} \lambda_{ij} q_{ij} = b$$

Teniendo el problema esta estructura, la forma más eficiente para abordarlo es por medio del simplex revisado, el cual nos proporciona la metodología adecuada para poder encontrar los valores de d_{kj} y f_{kj} a medida que se necesiten, sin tener que llevar a cabo toda la metodología del simplex tradicional para llegar a la solución óptima. La forma de generar las columnas implícitamente es lo que se conoce como *técnica de generación de columnas*, tema que será abordado posteriormente. De otra forma la transformación hecha para simplificar el problema no haría gran diferencia, sobre todo porque el número de variables aumenta con dicha transformación. El número de puntos extremos se desconoce y por lo tanto sería imposible encontrar los valores de d_{kj} y f_{kj} de manera explícita.

Para comenzar a explicar la metodología considérese *B* una matriz básica, ρ_B vector que contiene las variables en la base, f_B el vector de los coeficientes de costo de las variables de la base y $\pi = (\pi_1, \pi_2) = f_B B^{-1}$ vector de variables duales.

En la metodología del simplex un valor que determina la optimalidad o no de un problema o la existencia de variables que puedan mejorar el valor de la función objetivo es el que toman los $z_{kj} - f_{kj}$.

Así,

$$\begin{aligned} z_{kj} - f_{kj} = f_B B^{-1} q_{kj} - f_{kj} = \pi_1 d_{kj} + \pi_{2j} - f_{kj} &= \pi_1 (A_j x_{kj}^*) + \pi_{2j} - c_j x_{kj}^* \\ &= (\pi_1 A_j - c_j) x_{kj}^* + \pi_{2j} = z_j + \pi_{2j} \end{aligned} \qquad (3)$$

Como lo que se busca es encontrar soluciones factibles se debe analizar el problema cuando el mínimo de los $z_{kj} - f_{kj}$ sea negativo, es decir:

$$\text{Min}_{k,j} (z_{kj} - f_{kj})$$

Para resolverlo, tomando en cuenta la igualdad obtenida en (3) y el hecho de que el $\text{Min}_{k,j}$ $(z_{kj} - f_{kj})$ se encuentra en un punto extremo de las regiones comprendidas por $B_j x_j = b_j$ para toda *j*, la solución se encontrará resolviendo el problema planteado a continuación al que se le sumará el valor que se obtiene de π_{2j}.

$$\begin{aligned} & z_j = \min(\pi_1 A_j - c_j) x_j \\ & \text{s.a.} \\ & \quad B_j x_j = b_j \\ & \quad x_j \geq 0 \quad \text{para } j = 1, \ldots, r \end{aligned}$$

Ahora, el paso que se sigue en la metodología de la descomposición, consiste en encontrar soluciones óptimas a estos subproblemas encontrados. Dichos subproblemas sirven para generar posibles soluciones del problema maestro, es decir, mediante la solución de dichos problemas se determina la columna que entrará en la solución del problema maestro. Éste

proceso define de manera más clara lo que se conoce como un método de generación de columnas, ya que de manera implícita se determina la columna que entrará en la solución del problema maestro. Si las soluciones generadas por los subproblemas no son óptimas para el problema maestro, se plantean nuevos subproblemas a partir de la solución actual de éste y se busca su solución óptima. Éste proceso se sigue de forma iterativa hasta encontrar la solución óptima del problema maestro.

La relación que existe entre el problema maestro y los subproblemas se ilustra en la siguiente figura 2.

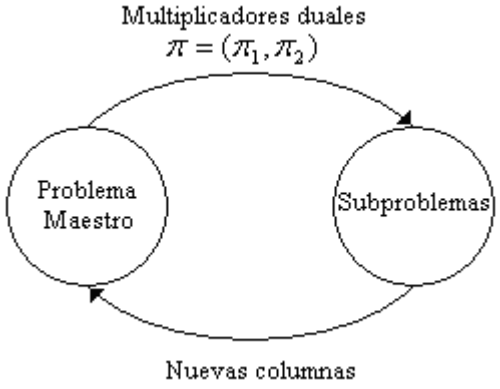


Fig. 2 Relación entre problema maestro y los subproblemas

Al encontrar los valores de z_j para toda j =1,…,r , por medio de las soluciones obtenidas se determinan los valores de $z_{kj} - f_{kj}$. Con el punto x_{kj}^* que se ha encontrado, ahora si se pueden determinar los valores d_{kj} y f_{kj} que formaran la columna que entrará al problema maestro. Cuando todos los $z_{kj} - f_{kj} \geq 0$, se cumple el criterio de optimalidad, de lo contrario si la solución no es óptima, se vuelven a establecer los subproblemas a partir de los nuevos valores de $\pi = (\pi_1, \pi_2)$ y se repite el proceso hasta encontrar una solución factible al problema maestro. Una vez encontrado éste, con el resultado obtenido se determinan ya los valores λ^*_{ij} y de x_{kj}^* que nos ayudarán a encontrar la solución del problema original.

2.1.3 Métodos de generación de columnas

La idea de la generación de columnas surgió por primera vez en 1958 cuando Ford y Fulkerson propusieron trabajar implícitamente con las columnas de un problema de flujo. Pero no fue sino hasta 1960 que Dantzig y Wolfe formalizaron este concepto aplicándolo a problemas de programación lineal que poseían una estructura que podía ser "descompuesta". Gilmore y Gomory demostraron posteriormente que ésta metodología podía ser aplicada con efectividad a problemas de optimización de cortes (Cutting Stock Problem). Actualmente la aplicación de éstas técnicas se ha extendido a varios campos de

la optimización combinatoria como lo son los problemas de organización de horarios y asignación de tripulaciones en compañías aéreas, entre muchos otros. (Desrosiers, 2003)

Los métodos de generación de columnas son métodos que se utilizan para resolver problemas de programación lineal y entera que tienen la particularidad de poseer un número de variables muy grande, es decir, problemas considerados de gran escala. Algunos autores consideran estos métodos como esquemas de costeo implícito de las variables no básicas para éste tipo de problemas.

Considere el siguiente problema denominado problema maestro:

$$\begin{aligned} Max\ z &= \sum_{j \in J} c_j \lambda_j \\ s.a.\ \sum_{j \in J} a_j \lambda_j &= b \\ \lambda_j &\geq 0, \quad j \in J \end{aligned}$$

Como los problemas con muchas variables dan lugar a problemas con matrices de coeficientes que contienen demasiadas columnas, esto genera por consecuencia tener que trabajar con matrices muchas veces imposibles de manejar debido al número de operaciones que involucra el trabajar con ellas o matrices difíciles de tratar, lo cual hace de la búsqueda de la solución óptima además de una tarea complicada, un procedimiento poco eficiente o imposible de realizar.

El método de generación de columnas consiste en trabajar con un subconjunto de las variables del problema maestro, formando con éste, otro problema denominado problema restringido, llamado así obviamente por limitarse el número de variables del problema original.

Como se recordará en cada iteración del método simplex se busca una variable no básica que mejore la solución de un problema de programación lineal, tomando en cuenta los costos reducidos. Debido a que en los problemas de los que hemos venido hablando el número de columnas es enorme, el determinar los costos reducidos asociados a las variables no básicas de éstos de manera explícita es imposible. Por lo que el método de generación de columnas trabaja con un subconjunto de las variables formando un problema restringido y más que determinar el costo reducido de cada variable y evaluarlo uno por uno, encuentra la variable no básica con el mejor costo reducido por medio de la solución de un conjunto de subproblemas.

Dicho de otra forma, el papel que juegan estos subproblemas es el de proveer una columna que pueda mejorar la solución del problema maestro o probar que no existe ninguna columna que lo haga. Así, en lugar de resolver un problema de enumeración implícita lo que se hace es resolver problemas de optimización, lo cual es mucho más sencillo. De esta forma se evita el tener que trabajar con toda la matriz de manera explícita. Sólo se llevan a cabo los cálculos necesarios para la variable que ingresará a la base.

El propósito del problema restringido por un lado es el de proveer los multiplicadores duales que se transfieren a los subproblemas y que determinan los coeficientes de costo de éstos y por otro lado controlan el criterio de paro. Sólo cuando ya se ha terminado con el proceso se ha encontrado una solución factible para el problema original.

Problema restringido:

$$Max\ z = \sum_{j \in J'} c_j \lambda_j$$

$$s.a.\ \sum_{j \in J'} a_j \lambda_j = b$$

$$\lambda_j \geq 0, \quad j \in J'$$

$$con\ J' \subset J$$

Los subproblemas también considerados subproblemas de generación de columnas o simplemente generadores de columnas tienen el propósito de identificar la columna con el menor costo reducido, de acuerdo al criterio de optimalidad que se ha elegido y a que el problema a optimizar se trate de un problema de minimización. Si el mínimo costo reducido es negativo, se ha encontrado una columna que ingresará a la base. De ésta forma el problema restringido recibe una nueva columna Y_{ij}. La cual está basada en los coeficientes de costo determinados por medio de los multiplicadores duales que proporcionó el problema restringido.
Si el mínimo costo reducido es mayor o igual que cero se ha probado que la solución actual del problema restringido es óptima para el problema maestro.

La principal ventaja de la generación de columnas es que no se necesitan determinar todas las columnas inicialmente o acarrear con ellas durante todo el proceso de solución. Éstas sólo se generan conforme se vayan necesitando.

2.2 Heurísticas

Debido a que el problema de la organización de horarios se encuentra dentro de los problemas denominados combinatorios y a que en muchas ocasiones es fácil encontrar soluciones factibles por simple inspección o por conocimiento de la estructura del

problema, a lo largo del tiempo diversos métodos heurísticos se han implementado para hallar la solución a los problemas ya mencionados.

Muchas veces el hallar la solución óptima a un problema combinatorio utilizando un método exacto implica mucho tiempo, ya que como se menciona en (Hincapié Isaza, *et. al.*, 2004) "Un gran problema de la optimización es el fenómeno llamado explosión combinatorial, que significa, que cuando crece el número de variables de decisión del problema, el número de decisiones factibles y el esfuerzo computacional, crecen en forma exponencial", por lo que las heurísticas proporcionan una solución a esto hallando resultados "satisfactorios" en un tiempo razonable.

Las heurísticas son métodos intuitivos que garantizan hallar una solución a un problema pero no necesariamente óptima e incluso esta solución puede ni siquiera ser buena.

La literatura clasifica los algoritmos heurísticos en: constructivos, es decir, que construyen una solución desde el inicio asignando valores a una o varias variables a la vez, algoritmos de descomposición y división, algoritmos de reducción, algoritmos de manipulación del modelo y algoritmos de mejora como los algoritmos de búsqueda local. Dentro de esta última clasificación se encuentran los Algoritmos Genéticos (AG), Recocido Simulado, Búsqueda Tabú (TS), Colonia de Hormigas (ACO) y GRASP.

Los algoritmos heurísticos utilizados para la solución de problemas de organización de horarios obtienen una solución inicial y después la mejoran basándose en la búsqueda en la vecindad como los de Búsqueda Tabú y los Algoritmos Genéticos principalmente. Por esta razón nos enfocaremos a estos tipos de heurísticas.

2.2.1 Algoritmos de búsqueda local

Los algoritmos de búsqueda local "Local Search algorithms" o también conocidos algoritmos de búsqueda en la vecindad "Neighborhood search algorithms", son considerados algoritmos de mejora. Esto implica que los algoritmos parten de una solución inicial y buscan encontrar una mejor en cada iteración examinando el vecindario de la actual solución.

Estos algoritmos obtienen una solución factible inicial ya sea de manera aleatoria o por medio de algún algoritmo. Buscan una solución vecina que mejore el valor de la función objetivo del problema hasta que ya no se encuentre alguna solución que mejore el valor de la función. En estos algoritmos se necesitan definir tres puntos importantes: primero cuál sería una solución, segundo una función de costo que determine si las soluciones vecinas son mejores o no en comparación con la solución actual y tercero una estructura de las vecindades de las soluciones. Un problema de estos algoritmos es que al no permitir tomar soluciones que no mejoren el valor de la función se puede incurrir en óptimos locales, lo que puede propiciar soluciones poco eficientes.
Para contrarrestar esta situación se han creado las conocidas como metaheurísticas que además de contar con elementos que evitan incurrir en óptimos locales son heurísticas que pueden ser aplicadas a una variedad más extensa de problemas.

A continuación se presentan características generales de los algoritmos de búsqueda más conocidos.

2.2.2 Recocido simulado

El método de recocido simulado se basa en un proceso metalúrgico llamado de recocido, el cual se utiliza para obtener materiales más resistentes, ya que se logró asociar conceptos del proceso metalúrgico con elementos de optimización combinatoria, dicha asociación se muestra en la figura 3.

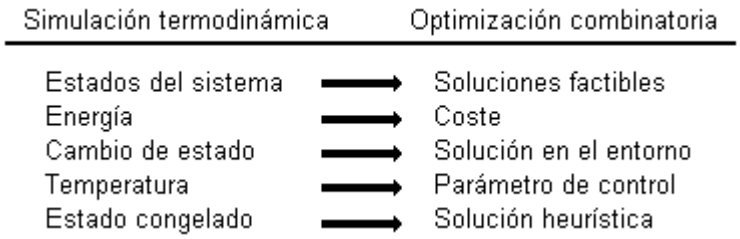


Fig. 3 Analogía de optimización combinatoria con proceso metalúrgico

El método de recocido consiste en calentar a muy alta temperatura el material para luego descenderla muy lentamente hasta que el material alcanza su máxima resistencia. Esto se logra cuando los átomos forman una estructura cristalina, es decir, cuando se ha alcanzado el estado de mínima energía.

El método de recocido simulado busca la solución óptima de un problema de manera análoga a este proceso.

Parámetros del modelo:

1. Temperatura inicial
2. Grado de enfriamiento
3. Criterio de finalización, es decir, cuándo se llega al estado de mínima energía.
4. Solución inicial

Como cualquier algoritmo de búsqueda, el algoritmo de recocido simulado inicia generando una solución factible inicial, ya sea de manera aleatoria o por medio de algún algoritmo, y estableciendo un valor inicial para el parámetro temperatura, el cual es un parámetro relacionado con una función de probabilidad que permitirá ciertos movimientos dentro del algoritmo.

A diferencia de los algoritmos de búsqueda el de recocido simulado sí permite considerar soluciones cuyo valor de la función objetivo pueda ser peor que el de la solución actual, es decir, si la solución que se obtiene es mejor que la actual se selecciona, si la solución es "peor" se selecciona con una probabilidad que depende del parámetro temperatura y el nivel de energía. Esta consideración lo que permite es escapar de óptimos locales, lo cual

es una ventaja sobre los algoritmos de búsqueda tradicionales. La forma en la que el algoritmo acepta las soluciones que se consideran "peores" lo logra de manera controlada mediante una función de probabilidad, que ya se había mencionado anteriormente, que hará disminuir la probabilidad de movimientos a ese tipo de soluciones conforme avanza la búsqueda.

Al inicio del algoritmo la probabilidad de aceptar soluciones "peores" es alta y conforme se avanza ésta probabilidad decrece gradualmente, es decir, como se explica en Dowsland *et. al.* (2001), "inicialmente casi todos los movimientos se aceptan, explorándose aleatoriamente sobre todo el espacio de soluciones. Conforme se avanza en el algoritmo y al decrecer la probabilidad, el aceptar nuevas soluciones será menor, por lo que al final del algoritmo, sólo los movimientos que mejoran la función se aceptarán".

2.2.3 Búsqueda Tabú

El método de búsqueda Tabú es un método que se basa en la búsqueda local o también conocida búsqueda por vecindades, que al igual que el método de recocido simulado tiene una forma particular de evitar incurrir en óptimos locales. Esto lo logra por medio de una "memoria" o "lista tabú" que almacena los últimos movimientos hechos durante la búsqueda, es decir, las últimas soluciones obtenidas y que se mantendrán allí por un cierto número de iteraciones. Al igual que el método de recocido simulado sí permite aceptar soluciones "peores", mientras no se encuentren relacionadas con movimientos tabú. Siempre que un movimiento se encuentre en la lista tabú éste será un movimiento "prohibido", es decir, no podrá ser considerado en la búsqueda.

El principio en el que se basa el modelo es como dice Glover (1997) "Una mala elección basada en estrategias proporciona más información que una buena elección al azar".

2.2.4 Algoritmos Genéticos

Los Algoritmos genéticos son algoritmos de búsqueda que se basan en la teoría de la evolución de Darwin. Como menciona Coello (1995) "Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno".

Mediante una analogía de los procesos evolutivos y de los mecanismos de selección como lo son la recombinación, la mutación, y la selección se encuentran nuevas generaciones, las cuales muestran una mayor "adaptabilidad" que sus antecesores. Es decir, lo que se intenta es construir soluciones nuevas con lo mejor de soluciones anteriores.

En esta técnica se debe definir una función de aptitud que indique qué tan buena o mala es una cierta solución. Dicha función es la función objetivo que se busca optimizar.

Como ya se mencionó un punto importante en el diseño de algoritmos de búsqueda local es la elección de la estructura de la vecindad, es decir, la forma en la que se va a definir ésta.

Es importante ya que de ella depende que las soluciones halladas en la búsqueda tengan una alta precisión o no.

Existen problemas que, por sus características, el tamaño de la vecindad de sus soluciones es muy grande, por lo que la búsqueda explícita sobre la vecindad se hace impráctica. Esto propicia que sea necesario que la búsqueda se realice sólo en un segmento de la vecindad o que la búsqueda se haga de manera implícita. Lo que da pie a la creación de algoritmos de búsqueda de gran escala. Según Ravindra K. et. al (2004) muchas técnicas exitosas utilizadas en la investigación de operaciones pueden ser vistas como algoritmos de búsqueda de gran escala. Por ejemplo, el método simplex se puede ver como un algoritmo de búsqueda local y el método de generación de columnas como un algoritmo de búsqueda local de gran escala.

Capítulo 3

Problema de organización de horarios

El problema de organización de horarios a empleados, es un problema que consiste en la construcción de patrones de trabajo para el personal de una empresa y en la asignación de estos, durante un periodo fijo de tiempo. Este problema se conoce en la literatura como "The Scheduling Problem", "Rostering" o "Employee Timetabling Problem" y está considerado dentro de los problemas de distribución de recursos que a su vez es un problema de optimización combinatoria.

3.1 Categorías de la organización de horarios

De manera más general el problema de la organización de horarios se clasifica en 5 categorías básicas las cuales corresponden al tipo de problema que resuelven, según Rodríguez (2006) estas son:

1. Cuando el interés se centra en la organización de los días libres del personal para poder diariamente satisfacer la demanda de servicios requerida.
2. Cuando se tiene el interés en la asignación de personal, que consiste en determinar exactamente el número de personas que realizarán cierta tarea.
3. Cuando la prioridad es la creación de patrones de trabajo que cubran las demandas con un mínimo de personal distribuyendo la carga de trabajo.
4. Cuando lo importante es crear patrones de trabajo que además de contemplar periodos de trabajo y días libres, determinen la categoría de trabajo que se realizará. Lo que se hace bajo el supuesto de que el personal no está calificado de igual manera.
5. Cuando en lugar de centrarse en el personal se tiene interés en la agrupación de jornadas de trabajo como lo es por ejemplo el organizar los vuelos aéreos que posteriormente serán asignados a ciertos pilotos.

El modelo básico de nuestro interés intenta minimizar el costo de la asignación de los horarios que cubren los requerimientos de servicio de las empresas sujeto a un conjunto de restricciones.

Dichas restricciones se clasifican en restricciones fuertes y restricciones suaves. Las primeras implican factibilidad y estas restricciones tienen que satisfacerse a cualquier costo por lo que se les denomina como restricciones fuertes. Por otro lado las restricciones suaves son restricciones que idealmente se desearían satisfacer pero que muchas veces se ignoran para poder generar una solución viable.

Algunos ejemplos de estas restricciones son: La carga de trabajo (donde se puede establecer un mínimo y un máximo con respecto al número de horas a trabajar), los turnos de trabajo consecutivos que se pueden laborar (estableciendo un mínimo, un máximo o incluso un número exacto de turnos), preferencias laborales de los trabajadores, los días libres (estableciéndose también un mínimo, un máximo o el número consecutivo de días

que se pueden tener libres), el tiempo libre entre turnos, los patrones de turnos y las demandas por categorías del personal (si es el caso), entre otras.

Debido a todas estas consideraciones antes mencionadas, este problema es un problema considerado multi-objetivo, es decir, que son varios los objetivos que se quieren alcanzar al resolver el problema y muchos los criterios que se deben tomar en cuenta cuando se quiera evaluar la calidad de la solución propuesta. Son varios los criterios de optimización, los cuales están sujetos a restricciones. Algunos de estos criterios son: el uso de los recursos, la satisfacción de preferencias específicas de los trabajadores y el cumplimiento de los arreglos contractuales o legislaciones vigentes.

3.2 Organización de horarios para enfermeras

En el caso particular de la asignación de horarios a enfermeras, el proceso involucra la construcción de horarios con deberes y la asignación de enfermeras a periodos de tiempo que se deben cubrir en un día, tomando en cuenta las restricciones particulares del caso. El objetivo es construir un horario que satisfaga los requerimientos de cobertura del hospital sobre el horizonte de planeación. La figura 3.2.1 ilustra una asignación de *n* enfermeras, a *t* turnos, durante un periodo de *d* días.

	Día 1					Día 2					Día 3					…					Día d				
	T_1	T_2	T_3	…	T_t	T_1	T_2	T_3	…	T_t	T_1	T_2	T_3	…	T_t	T_1	T_2	T_3	…	T_t	T_1	T_2	T_3	…	T_t
E_1	*				*			*				*													*
E_2		*				*								*				*					*		
E_3		*						*	*				*												
E_4	*							*					*	*		*				*			*		
E_5				*	*													*							
⋮												*	*	*			*					*		*	
								*	*										*			*		*	
E_n			*	*																*					*

Fig. 3.2.1 Asignación de enfermeras

Aunque de primera impresión parezca que el cumplir con este objetivo es una tarea fácil, esto no es así, hay muchos factores que se tienen que tomar en cuenta. Primero la formulación del problema de manera que represente lo más exactamente posible las circunstancias del hospital en cuestión y segundo la manera en la que ya habiéndose planteado el problema se buscará la solución de éste. Aunado a esto el tamaño del problema se ve fuertemente afectado por tres factores: primero el número de enfermeras a las que se les asignará un horario, segundo el número de días sobre los que se hará la asignación y tercero el número de turnos que se deben de contemplar por día.

De manera general, al problema de la asignación se le asocia una matriz cuyas filas corresponden a las tareas que se deben realizar, que en este caso son los turnos de cada día que se tienen que cubrir y cuyas columnas corresponden a los horarios alternativos que pueden ser asignados a las enfermeras.

Dichas columnas se conforman por ceros y unos, en donde lo unos representan los periodos que trabajará la enfermera y los ceros periodos que no laborará. Así, cada horario factible define una columna de la matriz.Vanhoucke(2005)

Como en los problemas de organización de horarios es imposible trabajar con todos los posibles horarios para cada enfermera ya que generalmente son muchos, una manera en la que se resuelven estos problemas es generando un subconjunto de dimensiones razonables de los horarios factibles para cada trabajador.

Al momento de diseñar un horario se tiene que tomar en cuenta que existen distintos tipos de horarios: los horarios de corto plazo, los de medio y los de largo plazo. El plazo se refiere al número de días que contempla el horario. Así, un horario de corto plazo puede ser un horario diario, uno de medio plazo contempla un periodo semanal, mensual o bimestral y por último uno de largo plazo puede ser trimestral, semestral, etc. La longitud temporal del horario que se desea construir dependerá de los propósitos específicos de planeación que se desean lograr.

La metodología presentada en este trabajo se centra en el problema de la generación de horarios en donde las asignaciones de tareas se hacen por periodos de 28 0 30 días. Esto da pie a un sistema de planeación más flexible que permite considerar requerimientos individuales del personal.

Existen tres formas de resolver el problema de la organización de horarios en esta etapa de planeación, las cuales son:

1. Organización de horarios cíclica o rotacional
2. Auto organización (Self-scheduling)
3. Organización de horarios con preferencias

3.2.1 Organización de horarios cíclica o rotacional

La organización de horarios cíclica se centra por un lado en establecer un conjunto de horarios que en conjunto satisfagan las demandas del hospital, es decir, que cumplen con todos los requerimientos del periodo y por otro evitar patrones de trabajo indeseables, intentando minimizar el número de enfermeras requeridas por turno. Con este conjunto se forma un patrón que es repetido en los periodos posteriores, es decir, las enfermeras son rotadas de un conjunto de horarios a otro durante el periodo de planeación. Sólo se necesitará crear un nuevo horario cuando ocurra algún cambio en la demanda promedio de personal. Una de las ventajas de este tipo de horarios es que garantizan estabilidad a las enfermeras, ya que da oportunidad a que el personal de enfermería pueda organizar sus actividades personales. Por otro lado, tiene como desventaja la falta de flexibilidad ya que no contempla requerimientos de días libres y no toma en cuenta las preferencias individuales de las enfermeras por ciertos turnos de su elección. Por lo que su aplicación es limitada. La figura 3.2.1.1 muestra un ejemplo de asignación de horarios cíclica.

	Semana 1							Semana 2							Semana 3							Semana 4						
E1	N	N	N	N	N	-	-	-	-	T	T	T	T	T	T	T	-	-	M	M	M	M	M	M	-	-	N	N
E2	M	M	M	-	-	N	N	N	N	N	N	N	-	-	-	-	T	T	T	T	T	T	T	-	-	M	M	M
E3	T	T	-	-	M	M	M	M	M	M	-	-	N	N	N	N	N	N	N	-	-	-	-	T	T	T	T	T
E4	-	-	T	T	T	T	T	T	T	-	-	M	M	M	M	M	M	-	-	N	N	N	N	N	N	N	-	-

M : Turno de la mañana.
T :Turno de la tarde.
N :Turno de la noche.

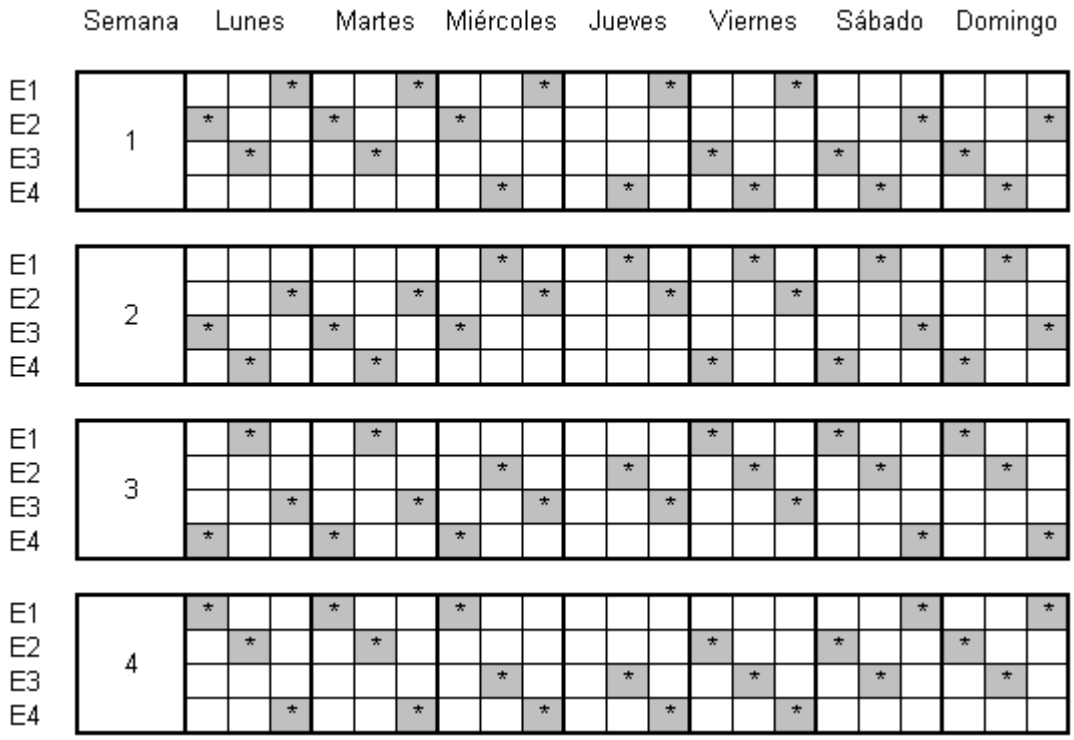


Fig. 3.2.1.1 Asignaciones de horarios cíclicos

3.2.2 Auto organización (Self-scheduling)

El método “self-scheduling” es un método manual mucho más flexible que la organización de horarios cíclica. El método consiste en primero determinar el mínimo número total de enfermeras que se requieren por turno cada día y establecer límites para el número máximo de enfermeras que se necesitan en cada uno para evitar excedentes. Después se solicita a las enfermeras que elijan los turnos en los que quieren trabajar. Todo este proceso es coordinado por la jefa de enfermeras o por el administrador de enfermería.
Esta metodología propicia que sea el personal de enfermería el que tenga el control de la generación del horario. Con esto son las propias enfermeras las que organizan sus horas de trabajo, lo que da más satisfacciones al personal y promueve la permanencia de este en la institución.

Las desventaja de este método es que la organización del horario sería muy difícil de hacer en condiciones en las que el número de enfermeras fuera muy grande, ya que al haber desacuerdos en la organización, sería poco probable que se pudiera llegar a consensos o arreglos. Esto provoca entre otras cosas problemas entre el personal, y puede propiciar una percepción de la organización del horario injusta o inequitativa o hasta de cierta manipulación por parte del encargado de la coordinación lo cual repercutiría en el ánimo del personal. Otro punto importante es que se desearía que las enfermeras tuvieran en mente mantener un equilibrio entre los beneficios individuales y los del hospital al elegir sus horarios y esto no siempre se considera.

El cuadro 1 muestra un ejemplo de una forma de registro para elegir el turno deseado por cada enfermera en base a los requerimientos del hospital.

Cuadro 1 Formato de registro

Semana 1

Hora	L	Iniciales	M	Iniciales	M	Iniciales	J	Iniciales	V	Iniciales	S	Iniciales	D	Iniciales
11:00 hrs.	3 G 2 E 1 A	*F. R.* *M. H* *L. E.*	3 G 3 E 2 A		3 G 1 E 1 A		2 G 2 E 2 A		4 G 1 E 1 A		3 G 5 E 3 A		5 G 2 E 1 A	
15:00 hrs.	4 G 2 E 2 A													
19:00 hrs.														
23:00 hrs.														
03.00 hrs.														
07:00 hrs.														

Tipo de enfermera

G: Enfermera general
E: Enfermera especialista
A: Enfermera auxiliar

G *Fernanda Rodríguez F.R.*
G *Martín Hernández M. H.*
A *Laura Estrada L.E.*

Fuente: ver referencia [1] .

[1] Cuadro basado en: Swansburg Russell C. y Richard J. Swansburg, Introduction to Management and Leadership for Nurse Managers, Tercera edición. Recuperado en : http://swansburg.jbpub.com/ppt/Chapter07.ppt

3.2.3 Organización de horarios con preferencias

La organización de horarios con preferencias es una metodología que combina los dos métodos anteriormente mencionados. Este método intenta incorporar la flexibilidad del método "self scheduling" manteniendo una parte de la estabilidad de los horarios cíclicos. Consiste en ajustar preferencias individuales de las enfermeras cuando se están elaborando los horarios. Las preferencias pueden ser medidas en términos de requerimientos de trabajar turnos específicos en días específicos o del otorgamiento de ciertos días libres, en términos de patrones de secuencias de turnos o del número de días consecutivos a trabajar. De igual forma que en "self-scheduling" se solicita a las enfermeras que se registren en turnos al inicio del horizonte de planeación y que proporcionen una lista de preferencias personales con respecto a sus horarios, si es que así lo desean. La jefa de enfermeras o el administrador de los servicios de enfermería se encargan de recopilar esta información y de tratar de satisfacer lo más posible las peticiones hechas (Bard 2005).

La organización de horarios con preferencias trata de equilibrar los requerimientos de los hospitales como lo son el proveer un servicio de calidad al menor costo posible con los requerimientos de las enfermeras que atienden a cuestiones individuales. Como ya se ha mencionado, estos requerimientos pueden ser preferir ciertos días libres, o el no desear ser asignadas a patrones de turnos pesados como sería el trabajar un turno de noche y al siguiente día tener que presentarse al turno de la mañana. Aunque este es un caso extremo, hay patrones de trabajo que no lo son tanto, pero que por la forma en la que están conformados pueden ser agotadores, lo que repercute en el desempeño del personal.

En general la función objetivo del problema minimiza la aversión de las enfermeras por ciertos patrones de trabajo, preferencias sobre los días libres y vacaciones, etc. Vanhoucke(2005).

3.2.3.4 Antecedentes

A través del tiempo se ha intentado resolver el problema de la organización de horarios con preferencias mediante la utilización de distintos métodos, como lo son: los métodos exactos como los basados en planos cortantes "Cutting Planes" y los basados en los métodos de ramificación y acotamiento "Branch and Bound", los métodos de satisfacción de restricciones y los basados en heurísticas. Esto ha dado pie a una infinidad de metodologías de solución al problema de la organización de horarios con preferencias.

Según Bard *et al.* (2005) algunos de estos métodos son:

1976 Warner	Fue el primero en desarrollar una metodología para resolver el problema planteado como "set covering" en el que todas las enfermeras trabajaban turnos de 8 horas. Utilizó una heurística denominada "greedy" para contrarrestar el tamaño del problema entero, en donde sólo se generaban 50 horarios factibles por enfermera.

1996 Berrada et al. Propusieron un modelo de satisfacción de restricciones que veía la cobertura de la demanda y el número de días de trabajo como restricciones fuertes, mientras que los patrones de turnos, los requerimientos diarios, días libres y fines de semana los veía como restricciones suaves.

1998 Jaumard et al. Extendió el modelo básico de programación entera al caso en el que se consideran turnos de 8 y de 12 horas. Adoptó el concepto de periodo de demanda para poder incluir turnos de 8 y de 12 horas simultáneamente en el modelo. Resolvió el problema modificado utilizando la descomposición de Dantzig y Wolfe con los ajustes necesarios para las restricciones de integralidad.

1998 Dowsland Desarrolló un método tabú con oscilación estratégica. El algoritmo comienza con un horario inicial obtenido con una heurística denominada "greedy" que ignora la mínima cobertura requerida y trata a cada enfermera separadamente.

2000 Valouxis and Housos
Combinaron la programación entera con heurísticas. En lugar de resolver el modelo clásico de set covering, resolvieron un modelo relajado que minimizaba el total de turnos necesitados para satisfacer la demanda sujeto a varias restricciones de preferencias.

2001 Kawanaka et al.
Usaron un algoritmo genético para hallar soluciones. Su modelo es restringido debido a que fue diseñado sobre seis restricciones fuertes específicas de un hospital.

En este trabajo se presenta un modelo y una metodología alternativa que explota el método de generación de columnas y las heurísticas para resolver el problema de la organización de horarios para enfermeras.

El modelo que se usará para resolver el problema es el propuesto en (Bard, 2005), que a continuación se describe:

3.3 Modelo de organización de horarios para enfermeras con preferencias[2]

Debido a que las restricciones del problema involucran el hecho de que los turnos de cada día deben de ser cubiertos por al menos un número específico de enfermeras, el problema se formula como un problema de cobertura "set covering".

[2] Bard, J.F. y H.W. Purnomo (2005), "European Journal of Operational Research", No. 164, pp. 510-534.

Antes de describir el modelo cabe mencionar que aunque la metodología contempla trabajar con turnos de 8 y de 12 horas simultáneamente, en este trabajo nos enfocaremos al caso en el que se consideran sólo turnos de 8 horas, es decir, tres turnos de trabajo por día (M: mañana, T: tarde y N: noche). La extensión al caso de turnos de 8 y de 12 horas sólo involucra el uso de los periodos de demanda. Al tener cinco tipos de turnos, tres de 8 horas y dos de 12 horas, con cuatro periodos basta para poder representar estos casos. Dos periodos de 8 horas y dos periodos de cuatro horas.

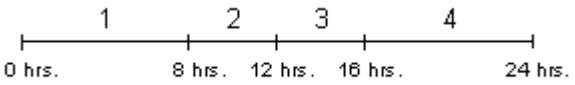


Fig. 3.3.1 Periodos de demanda para turnos de 8 y de 12 horas

Así, en lugar de tomar en cuenta la demanda por turno por cada día, se contempla la demanda por periodo de cada día, con lo que se evita traslapar horas de distintos turnos.

La representación en términos de periodos de cada uno de los turnos de 8 y de 12 horas de un día es la siguiente:

Turno	Duración	Periodos			
		1	2	3	4
Mañana	8 horas	1	0	0	0
Tarde	8 horas	0	1	1	0
Noche	8 horas	0	0	0	1
AM	12 horas	1	1	0	0
PM	12 horas	0	0	1	1

Fig. 3.3.2 Representación de los turnos de un día por medio de periodos de demanda

Para nuestro caso en el que contemplamos tres turnos de ocho horas el número de periodos coincide con el número de turnos. La razón de no contemplar en este trabajo turnos de 8 y de 12 horas simultáneamente es que esta extensión no es de relevancia para el desarrollo de la metodología.

3.3.1 Formulación del modelo:

El modelo básico minimiza el costo de satisfacer la demanda de una categoría particular de enfermeras en cada uno de los turnos por día del horizonte de planeación.

El modelo es el presentado a continuación:

$$Min \sum_{i \in E} \sum_{j \in H_i} c_{ij} x_{ij}$$

$$s.a. \quad DI_{dt} \leq \sum_{i \in E} \sum_{j \in H_i} a_{jdt} x_{ij} \leq DS_{dt} \ \ para\ toda\ d \in D, t \in T$$

$$\sum_{j \in H_i} x_{ij} = 1 \quad para\ toda\ i \in E,$$

$$x_{ij} = 0\ o\ 1 \ \ para\ toda\ i \in E,\ j \in H_i .$$

Donde :

Índices y conjuntos

i	es el índice para las enfermeras; $i \in E$.
j	es el índice para los horarios factibles considerados para cada enfermera i ; $j \in H_i$.
t	Asignación de turno (porción del día); $t \in T$.
d	Día del horizonte de planeación; $d \in D$.
E	Conjunto de enfermeras de planta a asignarles un horario.
H_i	Conjunto de columnas que representan los horarios factibles considerados para la enfermera i.
D	Conjunto de días en el horizonte de planeación (normalmente 28 o 42).
T	Conjunto de turnos en un día.
c_{ij}	Penalización "costo" de asignar el horario j a la enfermera i (este valor es una función del número y la severidad de preferencias y violaciones a requerimientos de la enfermera i en el horario j).

Parámetros

a_{jdt}	Toma el valor de 1 si el horario $j \in H_i$ contiene el turno t el día d y que toma el valor de 0 si no es así.
DI_{dt}	Límite inferior de la demanda de enfermeras el día d en el turno t.
DS_{dt}	Límite superior de la demanda de enfermeras el día d en el turno t.

Variables de decisión

x_{ij}	Variable binaria que toma el valor de 1 si la enfermera i es asignada al horario j y 0 si no es así, y se refiere a los horarios que son factibles para la enfermera i en términos de requerimientos y preferencias.

Las restricciones del problema garantizan:

1. Que el número de enfermeras que se asignan por turno sea un número entre un rango determinado. Dicho rango se establece muchas veces tomando en cuenta el número promedio de enfermeras necesitadas para cubrir la demanda de un cierto periodo a partir de la experiencia del hospital.
2. Que a cada enfermera le sea asignado uno y sólo un horario.

Un problema que afrontan muchos hospitales es la insuficiencia de cobertura que se origina por la escasez de personal, falta de enfermeras o fluctuación inesperada en la demanda. Esto lo resuelven de tres formas distintas: contratando enfermeras a través de agencias por períodos de 12 semanas, contratando enfermeras por un número específico de horas o reasignando enfermeras de unidades que tienen un exceso de personal en un periodo particular a unidades en donde hace falta personal.

El modelo que representa mejor esta situación es:

$$Min \sum_{i \in E} \sum_{j \in H_i} c_{ij} x_{ij} + M \sum_{d \in D} \sum_{t \in T} y_{dt}$$

$$s.a. \quad -s_{dt} + \sum_{i \in E} \sum_{j \in H_i} a_{ijdt} x_{ij} + y_{dt} = DI_{dt} \ \ para\ toda\ d \in D, t \in T$$

$$\sum_{j \in H_i} x_{ij} = 1 \quad para\ toda\ i \in E,$$

$$x_{ij} = 0\ o\ 1 \ \ para\ toda\ i \in E,\ j \in H_i .$$

$$y_{dt} \geq 0 \quad para\ toda\ d \in D, t \in T$$

$$0 \leq s_{dt} \leq DS_{dt} - DI_{dt} \quad para\ toda\ d \in D, t \in T$$

Donde :

y_{dt}	Número de enfermeras externas utilizadas el día *d* en el turno *t.*
s_{dt}	Variable de holgura concerniente al lado “izquierdo” de la restricción el día *d* en el turno *t.*
M	Número grande que representa el costo de emplear una enfermera externa.

El modelo intenta minimizar los costos de asignarle cierto horario a cada enfermera de planta y el costo por la utilización de enfermeras externas que cubran huecos en los turnos, ya sea por la falta de enfermeras o escasez de éstas. En la metodología presentada las enfermeras externas son utilizadas para cubrir turnos que quedan vacíos al intentar asignarles a las enfermeras de planta horarios que satisfagan lo mejor posible sus preferencias. Para cubrir estos huecos el modelo emplea la variable y_{dt} que corresponde a estos tipos de enfermeras.

Un punto importante en la formulación del modelo es la forma en la que se determinan los coeficientes de costo. Para que una metodología de solución para el problema de la organización de horarios con preferencias sea eficiente debe de encontrar una forma de construir los coeficientes de costo o también llamados de penalización de manera tal que valores pequeños de estos impliquen buenos horarios.

Así, en la metodología presentada, el costo de la asignación de cierto horario *j* a determinada enfermera *i*, c_{ij} se determina al definir este costo como una función creciente del número y grado de severidad de las violaciones hechas a las preferencias de la enfermera al asignarle dicho horario. Al depender este costo de las violaciones de las enfermeras, mientras se minimiza el costo de las asignaciones se intenta maximizar las preferencias de éstas. De esta forma, un buen horario será aquél cuyas violaciones a preferencias y requerimientos de las enfermeras sean mínimas.

La función propuesta para esta metodología, es la función exponencial $c_{ij}(v_{ij}) = 2^{v_{ij}-1}$. En donde v_{ij} cuantifica las violaciones hechas en el horario *j* a las preferencias de la enfermera *i*. Estas corresponden al número y grado de severidad de las violaciones en las que incurre el horario.

El costo M que se asigna al empleo de enfermeras externas se toma como: $M >> \max\{c_{ij} : i \in E, j \in H_i\}$ para de alguna forma limitar lo más posible este tipo de enfermeras.

Como se puede observar otra modificación que sufre el modelo es con respecto a la restricción uno del modelo básico. Como ya se había mencionado esta restricción se asegura de que el número de enfermeras asignadas diariamente por turno satisfaga los requerimientos sobre el número de enfermeras necesitadas para cubrir cada uno. Habiéndose establecido de antemano límites máximos y mínimos en la demanda de enfermeras por cada turno en el modelo básico se tenía la restricción:

$$DI_{dt} \le \sum_{i \in E} \sum_{j \in H_i} a_{jdt} x_{ij} + y_{dt} \le DS_{dt} \quad para\ toda\ d \in D, t \in T \quad (1).$$

Para simplificar el modelo se utiliza sólo la parte izquierda de ésta y se resta una variable de holgura denotada por s_{dt} en cada periodo del día en cuestión. Con lo que se obtiene la siguiente restricción:

$$-s_{dt} + \sum_{i \in E} \sum_{j \in H_i} a_{jdt} x_{ij} + y_{dt} = DI_{dt} \quad para\ toda\ d \in D, t \in T$$

Para asegurar que la restricción (1) se satisfaga en su totalidad las variables s_{dt} deben de satisfacer:

$$0 \le s_{dt} \le DS_{dt} - DI_{dt} \quad para\ toda\ d \in D, t \in T \qquad (2)$$

Que surge de la restricción:

$$DI_{dt} \leq \sum_{i \in E} \sum_{j \in H_i} a_{jdt} x_{ij} + y_{dt} \leq DS_{dt}$$

Y de restar DI_{dt} de la desigualdad

$$0 \leq \sum_{i \in E} \sum_{j \in H_i} a_{jdt} x_{ij} + y_{dt} - DI_{dt} \leq DS_{dt} - DI_{dt}$$

Las asignaciones se hacen sobre todas las clases de enfermeras del hospital, que en general son las enfermeras especializadas, las auxiliares de enfermeras y las enfermeras técnicas.

Las diferencias entre estos tipos de enfermeras radican en los diferentes rangos de cuidado que pueden proporcionar, en las responsabilidades que cada una posee y en la experiencia laboral que se tenga. Pero el modelo se ha restringido para resolver el problema para una sola categoría a la vez.

Capítulo 4

Metodología de solución

Al plantear el problema como uno de cobertura éste adquiere una complejidad particular por el tamaño del problema entero que se debe de resolver. Esto es lo que ha limitado un poco el empleo de la programación entera como forma de resolver este problema ya que el tamaño de éste depende del número de enfermeras, del número de turnos por día y del número de días sobre horizonte de planeación que se está contemplando. El número de horarios potenciales para cada enfermera crece de manera exponencial con el número de turnos y el número de días, por lo que el encontrar una forma para manejar el tamaño del problema es de vital importancia.

La metodología que se describe intenta contrarrestar esta situación al trabajar con un subconjunto de las variables del problema, o lo que se conoce como utilizar métodos de generación de columnas. Es decir, lo que se busca es generar solamente horarios (columnas) que satisfagan las restricciones fuertes y que propicien pocas violaciones a las preferencias de las enfermeras.

Cabe mencionar que la técnica de generación de columnas utilizada en la metodología propuesta es una técnica que atiende a las necesidades particulares del problema en cuestión. Esta técnica, como podrá observarse con posterioridad, no está vinculada a la mencionada en el capítulo 2 de éste trabajo la cual es una técnica general empleada en la solución de problemas de gran escala.

4.1 Metodología

La metodología propuesta en este trabajo inicia suministrando de información al modelo. Esta información es: el número de enfermeras de planta, la longitud del horizonte de planeación, el número de turnos, la demanda por turno y un conjunto de reglas inherentes al hospital que deben satisfacerse, ya sea por acuerdos contractuales, restricciones legales o políticas institucionales como lo son por ejemplo: el número de días consecutivos a laborar o el número de días libres consecutivos. Dichas reglas son divididas según su importancia en restricciones suaves y restricciones fuertes.

Al haberse determinado las demandas mínima y máxima de cada turno durante todos los días que cubren el horizonte de planeación, se solicita que todo el personal de planta se registre en algún turno de cada uno de los días del horizonte que se ha contemplado cubriendo el número de horas a laborar de acuerdo a su contrato.

Con este registro se crea un horario que se denominará horario base. Cabe mencionar que si la demanda, los requerimientos y las preferencias se satisfacen con este horario, no habría nada más que hacer. Pero lo más probable es que muchas de las enfermeras no se puedan registrar en el horario de su predilección propiciando violaciones a las preferencias. Esto se puede suscitar debido a que, aunque se les pide a las enfermeras que elijan horarios, habrá unas que se registren antes que otras lo que puede propiciar dicha situación.

Al haberse obtenido el horario base el siguiente paso consiste en evaluar la optimalidad de éste. Esto incluye verificar que las demandas de personal sean satisfechas, que no se viole ninguna restricción fuerte y que no se violen preferencias. Si éste es el caso, el horario base es óptimo y es el que se reportará como horario final. De no ser así se cuantifican las violaciones en las que se incurre para cada enfermera si ésta fuera la asignación.

El paso que sigue consiste en la generación de columnas e inicia cuando el horario base no es óptimo por cualquier incumplimiento de las condiciones antes mencionadas. Lo que se busca al generar columnas es proveer al modelo de las alternativas suficientes para generar una mejor solución en términos de los horarios para cada enfermera. Lo que implica construir horarios con pocas violaciones a preferencias, es decir, con coeficientes de costo bajos y que además eviten en lo posible el empleo de enfermeras externas.

Antes de continuar con la descripción de la metodología cabe mencionar un punto importante. Una de las componentes de la metodología se enfoca en generar horarios factibles para cada enfermera a partir del horario base, es decir, el conjunto de horarios factibles para cada enfermera será distinto. De esta forma se descarta la posibilidad de considerar cualquier vector (cero-uno), con *d* x *t* componentes, como un horario.
Con esto aunque el problema sigue siendo grande se disminuye considerablemente su tamaño ya que como se ha mencionado conforme el número de enfermeras sea mayor, el horizonte de planeación sea más extenso y los turnos se incrementen, el problema de asignación se vuelve cada vez más complicado. La metodología presentada, propone con la generación de columnas una solución para manejar el número de variables del problema, manteniendo eficiencia en la búsqueda de la solución del problema.

Como ya se ha mencionado la metodología genera un subconjunto de todos lo posibles horarios para cada enfermera a partir del horario base. La generación de los horarios alternativos de cada enfermera se hace modificando uno o dos turnos de este horario.

Para limitar el número de horarios (columnas) que se generarán, se contemplan tres parámetros:

$V_i^{\max}(\alpha)$ Representa el máximo número de violaciones permitidas para la enfermera *i*.

$Maxcol(\alpha)$ Representa el máximo número de columnas por enfermera que se incluirán en el modelo.

Maxcoltot Máximo número de columnas totales que incluirán en el modelo.

Tanto $V_i^{\max}(\alpha)$ como $Maxcol(\alpha)$ se definen como funciones no-decrecientes de α, en donde α representa una iteración del proceso.

En el caso de $V_i^{\max}(\alpha)$, para cada enfermera puede tomar un valor diferente de acuerdo a su contrato, antigüedad o consideraciones particulares que tome la persona que genera los horarios. Un punto importante es que la optimalidad del problema está estrechamente relacionada con estos parámetros ya que controlan directamente el tamaño del modelo. Los valores de estos parámetros se determinan al inicio cuando se suministra de información al modelo.

El proceso consiste en generar estas alternativas intercambiando turnos de trabajo de las enfermeras en los que se haya identificado que se puede disminuir el personal asignado a laborar por turnos de descanso en los que se haya identificado que se puede aumentar el personal de acuerdo a las demandas mínima y máxima. Esto se logra usando una heurística de vecindad de intercambio. Para ello, primero se identifican, por un lado, para cada enfermera los turnos que tienen posibilidad de disminuir el número de personal y en donde además la enfermera esté registrada para realizar el trabajo y por otro los turnos en los que hay posibilidad de aumentar el número de personas asignadas a laborar y la enfermera en cuestión no esté registrada para trabajarlos. Dichos periodos se agrupan en dos conjuntos, que serán los conjuntos entre los que se llevaran a cabo los intercambios.

Debido a que la optimalidad del problema se ve fuertemente afectada por el número de columnas que se consideren, habrá veces en las que sea necesario utilizar un proceso distinto para la generación de columnas ya que el número de columnas generadas para determinada enfermera pueda ser insuficiente para mejorar la solución. En lugar de restringir a que los turnos que se incluyan en los conjuntos de intercambio sean turnos en los que se pueda aumentar o disminuir personal de acuerdo a la asignación “actual”, se tomarán todos los turnos que la enfermera tenga asignados a trabajar para realizar los intercambios con todos los turnos que tenga asignados a no trabajar. Es decir, cualquier 1 de la columna del horario será candidato a intercambiarse por cualquier 0 del horario.

La manera en la que se determinará uno u otro método será tomando en cuenta el número de columnas alternativas de cada enfermera en el modelo y el número de columnas promedio.
Cuando el número de columnas totales de cierta enfermera exceda al número promedio de columnas por enfermera, se usará el primer método. Cuando éste número no exceda al número promedio se usará el método que no restringe la entrada a los turnos a los conjuntos de intercambio.

Para clarificar el proceso de intercambio se hará uso del siguiente ejemplo:

Supóngase que cierta enfermera tiene el siguiente horario para un horizonte de planeación de 5 días y tres turnos por día, es decir que contempla sólo turnos de 8 horas:

(100 |100 |100 |100 | 001),

que equivale a trabajar el primer periodo de los primeros cuatro días que contempla el horizonte de planeación y que corresponden al turno de la mañana y el quinto día trabajar el turno de la noche.

La metodología utiliza una heurística que genera horarios alternativos a éste intercambiando periodos activos (periodos asignados a trabajar) por no activos (periodos asignados como descanso).

En este caso dos posibilidades son:

1. Intercambiar el turno del último día para generar un patrón de trabajo de sólo turnos de la mañana. Es decir, pasar de un patrón (M M M N) a uno (M M M M).
   (100 |100 |100 |100 |100)

2. Por otro lado pasar de un patrón (M M M N) a uno (M M M T).

   (100 |100 |100 |100 | 010)

M: Turno de la mañana.
T: Turno de la tarde.
N: Turno de la noche.

Con esto se cuenta con tres horarios factibles para la enfermera en cuestión. Lo que amplia la posibilidad de mejorar la solución cuando se busque resolver el problema de asignación.

Hay que recalcar que sólo se permiten intercambios que no violen las restricciones de factibilidad del hospital y de cada enfermera. Así, aunque un periodo se encuentre dentro de alguno de los conjuntos, si el intercambio con otro periodo viola alguna restricción, no se crea la columna dada a partir de dicho intercambio. Es decir sólo se permite un intercambio siempre y cuando éste no produzca como consecuencia la violación a alguna restricción fuerte. Cuando ya se tienen las columnas alternativas se verifica si éstas no violan restricciones particulares de la enfermera, con lo que se determina si finalmente la columna es factible para ella o no.

Después de que se ha verificado que los horarios obtenidos son factibles, se determina el costo de asignar cada horario a la enfermera de acuerdo a las violaciones suaves en las que incurra éste.

Cabe mencionar que en la organización de horarios con preferencias una de las medidas para determinar la calidad de un horario es qué tan justo lo consideran las enfermeras con respecto a horarios asignados anteriormente. Lo que se busca en la metodología presentada es que cada vez que se inicie un proceso de asignación de horarios, ésta asignación, sea mejor que la anterior. Es decir, si a una enfermera se le asignó en un cierto mes un horario “malo”(con más violaciones que las deseadas), se trata de garantizar que en el siguiente mes le sea asignado un horario “bueno”(horario con pocas violaciones o al menos que el número de violaciones sea menor que en el horario anterior). Cabe recordar que la

asignación de horarios se hace cada determinado periodo, ya sea cada mes, cada dos meses, etc.

Esto se logra al limitar el número de violaciones de los horarios generados por lo que los horarios pasan por otro proceso de selección. Se considera el mínimo entre $V_i^{\max}(\alpha)$ parámetro establecido al inicio de la metodología y $V_i^{\max}$. Para determinar el valor de $V_i^{\max}$ se requieren el número de violaciones en los que incurrieron los horarios asignados a cada enfermera en el periodo anterior (semana, mes, bimestre, etc.), los cuales se denotarán por V_i^{ant}.

Para garantizar que se mejoraran los horarios que se asignarán a las enfermeras en el periodo actual con respecto al periodo de planeación anterior, se consideran la media ($\bar{v}$) y la desviación estándar ($\bar{\sigma}$) de las violaciones en que se incurrió en la asignación de horarios de todas las enfermeras en el periodo de organización anterior.

Para iniciar, cada valor $V_i^{\max}$ para cada enfermera se establece como infinito. Después cada valor V_i^{ant} es comparado con el valor obtenido de la media más la desviación estándar. Si V_i^{ant} es mayor que la suma, se establece $V_i^{\max} = \lfloor \max\{1, \bar{v} - \bar{\sigma}\} \rfloor$. Si esto no es así se deja el valor inicial y se prosigue a obtener el valor para la siguiente enfermera.

Así, después de determinar estos parámetros para cada enfermera, si el número de violaciones de cierto horario generado es mayor al máximo número de violaciones permitidas para una cierta enfermera, se descarta esa columna como horario alternativo para la enfermera en cuestión.

De los horarios que no fueron descartados se eligen aleatoriamente sólo un número de ellos. En la implementación de la metodología se describirá la técnica utilizada para la elección aleatoria de estos horarios. El objetivo de esta selección es controlar el tamaño del problema. Este número está en función del número de iteraciones y se establece al inicio de la planeación. Se denotó anteriormente por $Maxcol(\alpha)$.

Los horarios elegidos son revisados para verificar si son redundantes. Es decir, si ya se encuentran dentro del conjunto de horarios alternativos de la enfermera o no, ya que podrían duplicarse horarios. Para esto, se determina un identificador para cada columna y éste se almacena en una matriz. Cada enfermera tiene una matriz en donde se van almacenando los identificadores de las columnas alternativas generadas de cada una. Cuando se genera una columna nueva se verifica que el identificador que se le asigna no se encuentre ya en esta matriz.

La asignación del identificador a la columna se realiza de la siguiente manera:

Se tomará en cuenta la siguiente notación:

a y *b*	Turnos intercambiados que dieron origen a la nueva columna.
c_{ij}	Costo asociado a la nueva columna.
Cols	Número de renglones de la matriz que almacenará los índices de las columnas.
T(d)	Tipo de turno asignado a la enfermera el día *d* del horizonte de planeación.
$\Psi(T(d))$	Número aleatorio asociado al tipo de turno asignado a la enfermera el día *d*.
$\phi(d)$	Número aleatorio asociado al día *d*.
$\hat{d}$	Último día del horizonte de planeación.
Columnind	Identificador para la nueva columna.
Regind	Índice renglón en la matriz asociado a la nueva columna.

El valor del identificador del horario se determina de la siguiente manera:

$$Columnind = \sum_{d=1}^{\hat{d}} \left[\psi(T(d)) + \phi(d)\right]\left[\psi(T(d+1)) + \phi(d+1)\right].$$

Los valores de los identificadores se van almacenando en la matriz conforme se van creando, uno después de otro.

Una vez que se ha determinado que la columna es factible, que se ha calculado el costo y se ha determinado que el horario no es redundante, ésta se adhiere al conjunto de horarios alternativos para la enfermera. Cuando ya se han generado las columnas para todas las enfermeras se termina ésta etapa de la metodología.

Una vez generadas las columnas alternativas el siguiente paso consiste en actualizar la información del modelo, es decir, se adhieren las columnas generadas y se intenta resolver el problema de asignación.

4.2 Cuantificación de violaciones y determinación de coeficientes de costo.

Según el modelo que se está presentando el esquema de cuantificación de violaciones a preferencias o requerimientos es el mostrado en el cuadro 4.2.1. Con este esquema se intenta facilitar la penalización reduciendo esta labor a sólo tener que clasificar la violación en cualquiera de las cuatro categorías abajo mencionadas:

Cuantificación de violaciones a preferencias

Grado de la violación	Número de puntos de penalización por tipo de violación, v	Coeficiente de costo, $c_{ij}(v) = 2^{v-1}$
Simple	1	1
Seria	2	2
Severa	3	4
Extrema	4	8

Cuadro 4.2.1 basado en Bard (2005)

4.2.1 Ejemplo de cuantificación de violaciones:

Suponga que un horario de una cierta enfermera tiene la siguiente estructura para un horizonte de planeación de dos semanas:

L	M	M	J	V	S	D	L	M	M	J	V	S	D
M	M		N	N	N	M	M		T	T	T	T	T

M : Turno de la mañana.
T : Turno de la tarde.
N : Turno de la noche.

Supongamos que se consideran dos patrones de trabajo indeseables, los cuales han sido clasificados por la enfermera en cuestión de la siguiente manera:

1. Tener tres turnos nocturnos consecutivos el cual la enfermera clasifica como una violación seria.
2. Tener asignados a trabajar cinco días consecutivos que la enfermera clasifica como una violación simple.

De ésta forma el número de puntos de penalización por las violaciones en las que incurre el horario serían 4 y por lo tanto éste tendría un costo de $c(4) = 2^{4-1} = 8$. Ver figura 4.2.1.

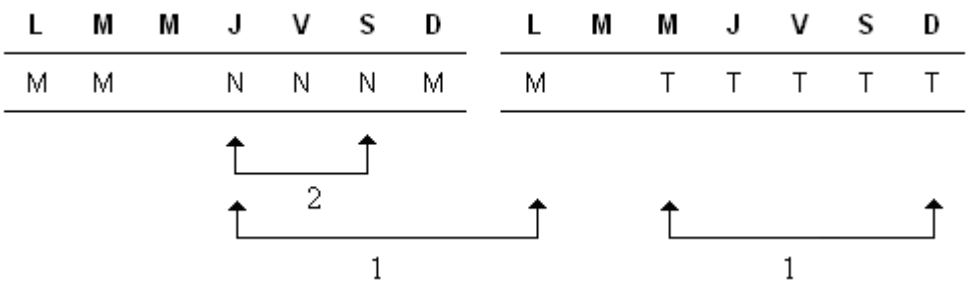


Fig. 4.2.1 Ejemplo de cuantificación de violaciones

Esta no es la única forma de cuantificar las violaciones a preferencias, al igual que se han propuesto modelos y métodos para resolver el problema, también se han propuesto diversas formas de cuantificación de violaciones, siendo algunas más simples que otras. Una es como la propuesta por Randhawa y Sitompul (1993), en donde sugieren que las enfermeras distribuyan 100 puntos entre todas las posibles violaciones, asignando mayor número de puntos a aquella violación que consideren más severa.

4. 3 Ejemplo

Supóngase que para un cierto hospital la información que se suministra es la mostrada en la figura 4.3.1

Número de enfermeras de planta a asignarles un horario	6
Días en el horizonte de planeación	4
Tipo de turnos	Turnos de 8 hrs.

Hora de inicio de los turnos

Turno	Hora de inicio
M	07:00 hrs.
T	15:00 hrs.
N	23:00 hrs.

Demanda de personal por turno

Turno	DI	DS
M	1	2
T	2	3
N	1	2

DI: Demanda inferior de enfermeras en el turno.
DS: Demanda superior de enfermeras en el turno.

Fig. 4.3.1

Maxcoltot = 2000

Máximo número de columnas por enfermera cada iteración=5

$\alpha^{\max}$ = máximo de iteraciones permitidas =10

Registro inicial de las enfermeras en turnos de su elección de acuerdo a la demanda de personal establecida:

Forma de registro

Turno	Hora de inicio	Lunes	Martes	Miércoles	Jueves
M	07:00 hrs.	*E2*	*E1* *E2*	*E1*	*E5*
T	15:00 hrs.	*E3* *E4* *E5*	*E5* *E6*	*E2* *E3*	*E3* *E6*
N	23:00 hrs.	*E6*	*E4*	*E4*	*E1*

Tomando en cuenta la definición de horario que se dio al inicio, el horario base que se obtiene a partir del registro de las enfermeras es el mostrado a continuación en la figura 4.3.2, en donde cada columna representa el horario que corresponde a cada enfermera a partir de su registro.

(d,t)	E_1	E_2	E_3	E_4	E_5	E_6
(1,1)	0	1	0	0	0	0
(1,2)	0	0	1	1	1	0
(1,3)	0	0	0	0	0	1
(2,1)	1	1	0	0	0	0
(2,2)	0	0	0	0	1	1
(2,3)	0	0	0	1	0	0
(3,1)	1	0	0	0	0	0
(3,2)	0	1	1	0	0	0
(3,3)	0	0	0	1	0	0
(4,1)	0	0	0	0	1	0
(4,2)	0	0	1	0	0	1
(4,3)	1	0	0	0	0	0

Fig. 4.3.2 Horario base

Para el ejemplo se considerarán las siguientes estructuras de penalización por violaciones:

Estructura de penalización por violaciones a preferencias del hospital

Restricciones	Restricción fuerte	Penalización por estar en el límite
No se debe trabajar más de un turno por día	√	
Días de trabajo consecutivos	≤ 3	1
Días libres consecutivos	≤ 2	1
Minreq	√	
Número de días que debe laborar cada enfermera	3	

Minreq: Representa peticiones personales de las enfermeras solicitadas al inicio de la planeación.
√: Se califica como restricción fuerte.

Supongamos que para el ejemplo sólo las enfermeras 3 y 4 realizan peticiones. La enfermera 3 requiere no trabajar el turno de la mañana y la enfermera 4 tener como día de descanso el cuarto día.

La estructura de penalización por violaciones a preferencias individuales de las enfermeras se muestra a continuación, la cual muestra el grado de aversión de éstas por los patrones de trabajo identificados como indeseables y los cuales se consideran como restricciones suaves.

Estructura de penalización por violaciones a preferencias individuales

Patrones de trabajo indeseables	E_1	E_2	E_3	E_4	E_5	E_6
001 100 *	1	1	2	3	2	1
000 100 000 ** 000 010 000 ** 000 001 000 **	1	1	1	2	1	1
Días de trabajo consecutivos igual a 3	1	3	1	2	2	2

* Patrón de trabajo que consiste en trabajar un día un turno de noche y al siguente un turno de mañana.

** Patrón de trabajo que consiste en tener un día de trabajo intercalado entre dos días libres.

Lo que se busca es de acuerdo a su aversión evitar horarios con los patrones:

$$(0\ 0\ 1\ |1\ 0\ 0)^T$$

$$(0\ 0\ 0|1\ 0\ 0|\ 0\ 0\ 0)^T$$

$$(0\ 0\ 0\ |0\ 1\ 0|\ 0\ 0\ 0)^T$$

$$(0\ 0\ 0\ |0\ 0\ 1|\ 0\ 0\ 0)^T$$

Se define α = 1.

Para ejemplificar la generación de columnas se considerará el horario de la enfermera 3 y se describirán los dos procesos con los cuales se forman los conjuntos de intercambio ya mencionados. Entonces, los conjuntos de intercambio que se generarán son los conjuntos A_3 y B_3. Para ello, se toma en cuenta el horario base y la demanda de personal por turno ya establecidos con anterioridad. Véase figura 4.3.3.

(d,p)	DI	E_1	E_2	E_3	E_4	E_5	E_6	DS
(1,1)	1	0	1	0	0	0	0	2
(1,2)	2	0	0	1	1	1	0	3
(1,3)	1	0	0	0	0	0	1	2
(2,1)	1	1	1	0	0	0	0	2
(2,2)	2	0	0	0	0	1	1	3
(2,3)	1	0	0	0	1	0	0	2
(3,1)	1	1	0	0	0	0	0	2
(3,2)	2	0	1	1	0	0	0	3
(3,3)	1	0	0	0	1	0	0	2
(4,1)	1	0	0	0	0	1	0	2
(4,2)	2	0	0	1	0	0	1	3
(4,3)	1	1	0	0	0	0	0	2

Fig. 4.3.3

Si se toma en cuenta la posibilidad de aumentar y disminuir el número de personal en los turnos se tienen los siguientes conjuntos:

A_3: Conjunto de periodos en donde se encuentra asignada a laborar la enfermera 3 y se puede disminuir el número de enfermeras asignadas en el turno sin afectar la demanda mínima de personal.

B_3: Conjunto de periodos en donde no se encuentra asignada a laborar la enfermera 3 y se puede aumentar el personal para laborar ese turno sin exceder la demanda.

Así, se tiene que:

$$A_3 = \{(1,2)\}$$
$$B_3 = \{(1,1),(1,3),(2,1),(2,2),(2,3),(3,1),(3,3),(4,1),(4,3)\}$$

Tomando en cuenta los conjuntos, los horarios originados por los posibles intercambios, en términos de periodos de descanso y periodos de trabajo, son los mostrados en la figura 4.3.4.

1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0	0
0	0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**	0
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	**1**

Fig. 4.3.4

En donde los unos y los ceros en negritas corresponden a los periodos intercambiados. Como se puede observar algunos de estos horarios no son factibles. Esto debido a que violan restricciones del hospital o restricciones concernientes a la enfermera 3. El horario 1 asigna a trabajar a la enfermera 3 el periodo 1 del día 1 lo que equivale a trabajar el turno de la mañana de ese día, lo que viola un requerimiento de la enfermera. Los horarios 6 y 7 asignan a trabajar a la enfermera 3 dos turnos el tercer día lo que viola una restricción del hospital. De igual forma los horarios 8 y 9 asignan a la enfermera 3 a trabajar dos turnos el cuarto día.

Así pues, los únicos horarios considerados para la enfermera 3 son los mostrados en la figura 4.3.5.

h_{32}	h_{33}	h_{34}	h_{35}
0	0	0	0
0	**0**	**0**	**0**
1	0	0	0
0	**1**	0	0
0	0	**1**	0
0	0	0	**1**
0	0	0	0
1	1	1	1
0	0	0	0
0	0	0	0
1	1	1	1
0	0	0	0

Fig. 4.3.5

Se denotarán con h_{ij} los horarios *j* factibles para la enfermera *i*. Es decir, los horarios que se irán agregando a los conjuntos H_i para cada enfermera.

Si se toman en cuenta todos los periodos de trabajo y todos los periodos libres de la enfermera, los conjuntos que se crean son:

A_3: Conjunto de periodos en donde se encuentra asignada a laborar la enfermera 3.
B_3: Conjunto de periodos en donde no se encuentra asignada a laborar la enfermera 3.

Así, se tiene que:

$$A_3 = \{(1,2),(3,2),(4,2)\}$$
$$B_3 = \{(1,1),(1,3),(2,1),(2,2),(2,3),(3,1),(3,3),(4,1),(4,3)\}$$

Tomando en cuenta los conjuntos, los horarios originados por los posibles intercambios, en términos de periodos de descanso y periodos de trabajo, son los mostrados en la figura 4.3.6.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	*0*	*0*	1	1	1	1	1	1	0	*0*	*0*	1	1	1	1	1	1	1
0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0
0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0
1	1	1	1	1	*0*	*0*	1	1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	1	1	1	1	1	*0*	*0*	1	1
0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0
1	1	1	1	1	1	1	*0*	*0*	1	1	1	1	1	1	1	*0*	*0*	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	**1**

Fig. 4.3.6

En donde los unos y los ceros en negritas corresponden a los periodos intercambiados. Cabe mencionar que en este proceso se permiten los intercambios dobles, es decir, que se toman en cuenta periodos de días en los que ya se había asignado un periodo a laborar, por lo que es necesario intercambiar ese uno por un cero, los cuales están marcados en cursivas.

Los únicos horarios considerados para la enfermera 3 al tomar en cuenta las restricciones de factibilidad son los mostrados en la figura 4.3.7.

h_{32}	h_{33}	h_{34}	h_{35}	h_{36}	h_{37}	h_{38}	h_{39}	h_{310}	h_{311}	h_{312}	h_{313}	h_{314}	h_{315}
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	0	0
0	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0	0
0	0	0	**1**	0	0	**1**	0	0	0	0	**1**	0	0
0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
1	1	1	1	**0**	**0**	**0**	**0**	**0**	1	1	1	1	1
0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
1	1	1	1	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**
0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Fig. 4.3.7

Después de que se verifica que los horarios obtenidos son factibles, se determina el costo de asignar cada horario a la enfermera de acuerdo a las violaciones suaves en las que incurra éste.

Siguiendo con el ejemplo, las columnas generadas a partir del horario base para cada enfermera en la iteración inicial son las mostradas a continuación, para esto, primero se supuso un máximo de columnas generadas por enfermera igual a cinco. Cuando el número de horarios generados es menor al parámetro establecido se consideran todos los horarios generados.

h_{32}	h_{33}	h_{34}	h_{35}	h_{42}	h_{43}	h_{52}	h_{53}	h_{54}	h_{55}	h_{56}
0	0	0	0	**1**	0	**1**	0	0	0	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	0	0	0	0	**1**	0	**1**	0	0	0
0	**1**	0	0	0	0	0	0	0	0	0
0	0	**1**	0	0	0	1	1	1	1	1
0	0	0	**1**	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	**1**	0	0
1	1	1	1	0	0	0	0	0	**1**	0
0	0	0	0	1	1	0	0	0	0	**1**
0	0	0	0	0	0	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Fig. 4.3.8 Columnas generadas a partir del horario base

Así, el problema de asignación de horarios que se tiene que resolver es:

MIN

$$2X_{11}+4X_{21}+0.5X_{31}+0.5X_{32}+2X_{33}+0.5X_{34}+0.5X_{35}+4X_{41}+2X_{42}+2X_{43}+4X_{51}+0.5X_{52}$$
$$+0.5X_{53}+2X_{54}+2X_{55}+4X_{56}+0.5X_{61}+50\sum_{d\in D}\sum_{t\in T}Y_{dt}$$

s.a.

$$-\mathbf{I}_{(12x12)}\begin{bmatrix}S_{11}\\S_{12}\\S_{13}\\S_{21}\\S_{22}\\S_{23}\\S_{31}\\S_{32}\\S_{33}\\S_{41}\\S_{42}\\S_{43}\end{bmatrix}+\begin{bmatrix}0&1&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0&1&0&0&1&0&0&0&1\\1&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0&0&1&1&1&1&1&1&1\\0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0\\1&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\0&1&1&1&1&1&1&0&0&0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0\\0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&1\\1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{bmatrix}\begin{bmatrix}X_{11}\\X_{21}\\X_{31}\\X_{32}\\X_{33}\\X_{35}\\X_{34}\\X_{41}\\X_{42}\\X_{43}\\X_{51}\\X_{52}\\X_{53}\\X_{54}\\X_{55}\\X_{56}\\X_{61}\end{bmatrix}+\mathbf{I}_{(12x12)}\begin{bmatrix}Y_{11}\\Y_{12}\\Y_{13}\\Y_{21}\\Y_{22}\\Y_{23}\\Y_{31}\\Y_{32}\\Y_{33}\\Y_{41}\\Y_{42}\\Y_{43}\end{bmatrix}=\begin{bmatrix}1\\2\\1\\1\\2\\1\\1\\2\\1\\1\\2\\1\end{bmatrix}$$

$$\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0\\0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\end{bmatrix}\begin{bmatrix}X_{11}\\X_{21}\\X_{31}\\X_{32}\\X_{33}\\X_{35}\\X_{34}\\X_{41}\\X_{42}\\X_{43}\\X_{51}\\X_{52}\\X_{53}\\X_{54}\\X_{55}\\X_{56}\\X_{61}\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\\1\\1\end{bmatrix}$$

$0 \le S_{dt} \le DS_{dt} - DI_{dt}$, para toda $d \in D$ y $t \in T$, donde $D = \{1,2,3,4\}$ y $T = \{1,2,3\}$

$Yt \ge 0$, para toda $d \in D$ y $t \in T$

$X_{11}, X_{21}, X_{31}.X_{32}, X_{33}, X_{34}, X_{41}, X_{42}, X_{51}, X_{52}, X_{53}, X_{54}, X_{55}, X_{61} \ge 0$

En este punto se pueden presentar varios escenarios:

1. Que no se encuentre una solución y que el número de columnas del modelo sea menor que el máximo de columnas totales permitidas en el modelo (*Maxcoltot*), con lo que se volvería a generar columnas para cada enfermera con el mismo horario base.
2. Que no se encuentre una solución y que el número de columnas del modelo sea mayor o igual que *Maxcoltot* , con lo que se reportaría el último horario como horario final.
3. Que se tenga una solución. Entonces se actualiza el horario base de las enfermeras tomando la solución obtenida en la asignación. Si el número de columnas del modelo es menor que *Maxcoltot* y el número de iteraciones es menor que el máximo de iteraciones permitidas, esto da lugar a dos posibilidades.

    a) Si el número de enfermeras externas es positivo se realiza otra iteración comenzando por generar nuevas columnas a partir del horario base actualizado.
    b) Si el número de enfermeras externas es igual a cero se eliminan todas las variables de enfermeras externas del modelo y se generan nuevas columnas a partir del horario base actualizado.

    Si el número de columnas del modelo es mayor que *Maxcoltot* y el número de iteraciones es mayor o igual que el máximo de iteraciones permitidas, se reporta el último horario obtenido como horario final.

Para el ejemplo, al resolver el problema de asignación de la primera iteración, se presentó el tercer escenario. La solución que se obtuvo fue la siguiente:

$$Z = 11.5$$

X_{11}	=	1	S_{11}	=	1	Y_{11}	=	0
X_{21}	=	1	S_{12}	=	0	Y_{12}	=	0
X_{31}	=	1	S_{13}	=	0	Y_{13}	=	0
X_{32}	=	0	S_{21}	=	1	Y_{21}	=	0
X_{33}	=	0	S_{22}	=	0	Y_{22}	=	0
X_{34}	=	0	S_{23}	=	0	Y_{23}	=	0
X_{35}	=	0	S_{31}	=	0	Y_{31}	=	0
X_{41}	=	1	S_{32}	=	0	Y_{32}	=	0
X_{42}	=	0	S_{33}	=	0	Y_{33}	=	0
X_{43}	=	0	S_{41}	=	0	Y_{41}	=	0
X_{51}	=	0	S_{42}	=	0	Y_{42}	=	0
X_{52}	=	1	S_{43}	=	0	Y_{43}	=	0
X_{53}	=	0						
X_{54}	=	0						
X_{55}	=	0						
X_{56}	=	0						
X_{61}	=	1						

Como el máximo de columnas en el modelo y el máximo de iteraciones permitidas son menores a las cotas establecidas, entonces se hace $\alpha = 2$. Se establece la asignación obtenida como el nuevo horario base y a partir de él se generan nuevas columnas para intentar obtener una mejor solución.

4.4 Implementación

La implementación de la metodología presentada se hizo en MATLAB versión 7.3.0.267 (R2006b). Se utilizó la caja de herramientas de optimización (Optimization Toolbox) versión 3.1 (R2006b) para resolver los problemas enteros que se crean en cada iteración. Los cálculos fueron procesados en una computadora personal con procesador Intel Pentium III con 797 MHz.

Debido a que la caja de herramientas de optimización de MATLAB sólo resuelve problemas binarios o lineales, las variables enteras del modelo se expresan en términos de variables binarias para poder resolver los problemas. La función de la caja de herramientas de optimización (Optimization Toolbox) de MATLAB que permite resolver problemas binarios es la función “bintprog”. La metodología utilizada por la función para resolver los problemas es la de ramificación y acotamiento “Branch and Bound”. La estrategia utilizada para elegir la variable sobre la que se va a ramificar se denomina en el programa por ‘maxinfeas’ y consiste en elegir la variable cuyo valor sea el más cercano a 0.5. La estrategia utilizada para elegir el nodo que se va a explorar es la denominada búsqueda del mejor nodo “best node search” que consiste en elegir el nodo que proporcione el valor más pequeño de la función objetivo. Todas las opciones de optimización que el programa permite modificar se dejaron con los valores predeterminados por el programa. Estas opciones son: el número de iteraciones, el número de nodos a explorar, la estrategia de ramificación y la de selección del nodo a explorar.

La selección aleatoria de los horarios generados durante la generación de columnas se implementó de la siguiente manera: Se utilizó la función “randperm(n)” de MATLAB, la cual genera una permutación de los enteros 1,2,…, n. El valor de n se determinó a partir del número de columnas que se generaron en la iteración correspondiente. Al obtenerse la permutación se eligen los primeros t números de la permutación. El valor de t es igual al mínimo entre el máximo de columnas permitidas por enfermera y el número de columnas que se generaron.

La implementación de la metodología se realizó en base a los algoritmos presentados en Bard(2005).

A partir de la metodología antes descrita se implementó el programa denominado **sched2**. A continuación se muestran algunos resultados obtenidos al utilizar el programa para resolver problemas de asignación.

Aplicación 4.4.1

El objetivo de esta aplicación es mostrar los conjuntos de horarios factibles que se generaron para cada enfermera en un problema de asignación de horarios. Como ya se mencionó lo que se busca es dotar al modelo de las alternativas suficientes que le permitan a la metodología hallar una "buena" solución.

Se consideraron 6 enfermeras de planta, un horizonte de planeación de 4 días, un día libre como máximo para cada enfermera, un número máximo de iteraciones igual a 5, un máximo de 6 columnas a generar por enfermera en cada iteración.

Demanda de personal:

Turno	min	máx
D	1	2
T	2	3
N	1	2

Sólo las enfermeras 1 y 4 hicieron solicitudes en cuanto a sus horarios, ambas solicitaron tener como día libre el segundo día del horizonte de planeación.

El horario que se suministró para iniciar el proceso es el ilustrado a continuación:

Horario base

Enfermera	Día			
	1	2	3	4
1	_	T	D	D
2	D	D	T	_
3	N	_	T	T
4	T	N	N	_
5	T	T	_	N
6	N	D	_	T

Los horarios que se fueron generando en cada iteración y que dan lugar a los conjuntos S_i para cada enfermera *i* se muestran a continuación. Los conjuntos resultantes en la quinta iteración no se mostraron ya que no hubo modificaciones con respecto a los generados en la cuarta iteración.

S_1

$\alpha = 1$ $\alpha = 2$ $\alpha = 3$ $\alpha = 4$

0		0	0	0	1		0	0	0	1	0	0	0		0	0	0	1	0	0	0		0	0	0	1	0	0	0
0		0	1	0	0		0	1	0	0	1	1	1		0	1	0	0	1	1	1		0	1	0	0	1	1	1
0		0	0	1	0		0	0	1	0	0	0	0		0	0	1	0	0	0	0		0	0	1	0	0	0	0
0		0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
1		1	0	0	0		1	0	0	0	0	0	0		1	0	0	0	0	0	0		1	0	0	0	0	0	0
0		0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
1		1	1	1	1		1	1	1	1	1	1	0		1	1	1	1	1	1	0		1	1	1	1	1	1	0
0		0	0	0	0		0	0	0	0	0	0	1		0	0	0	0	0	0	1		0	0	0	0	0	0	1
0		0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
1		1	1	1	1		1	1	1	1	0	0	1		1	1	1	1	0	0	1		1	1	1	1	0	0	1
0		0	0	0	0		0	0	0	0	1	0	0		0	0	0	0	1	0	0		0	0	0	0	1	0	0
0		0	0	0	0		0	0	0	0	0	1	0		0	0	0	0	0	1	0		0	0	0	0	0	1	0

S_2

$\alpha = 1$ $\alpha = 2$ $\alpha = 3$ $\alpha = 4$

1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0		1	0	0	0	0	0		1	0	0	0	0	0		1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0		0	0	0	0	1	0		0	0	0	0	1	0		0	0	0	0	1	0	1	1	1	1	1
0	1	0	0	0	0		0	1	0	0	0	0		0	1	0	0	0	0		0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1	0	0	0	1	1
0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0		0	0	1	0	0	0		0	0	1	0	0	0		0	0	1	0	0	0	0	1	0	0	1
0	0	0	1	0	0		0	0	0	1	0	0		0	0	0	1	0	0		0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	0	1		0	0	0	0	0	1		0	0	0	0	0	1		0	0	0	0	0	1	0	0	0	0	0

S_3

$\alpha = 1$ $\alpha = 2$ $\alpha = 3$ $\alpha = 4$

0	0	0	1	0		0	0	0	1	0	0		0	0	0	1	0	0	1	0	0		0	0	0	1	0	0	1	0	0
0	1	0	0	0		0	1	0	0	0	0		0	1	0	0	0	0	0	0	0		0	1	0	0	0	0	0	0	1
1	0	0	0	0		1	0	0	0	0	0		1	0	0	0	0	0	0	0	0		1	0	0	0	0	0	0	0	0
0	0	0	0	0		0	0	0	0	0	1		0	0	0	0	0	1	1	1	1		0	0	0	0	0	1	1	1	1
0	0	1	0	0		0	0	1	0	0	0		0	0	1	0	0	0	0	0	0		0	0	1	0	0	0	0	0	0
0	0	0	0	1		0	0	0	0	1	0		0	0	0	0	1	0	0	0	0		0	0	0	0	1	0	0	0	0
0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1
0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	1		0	0	0	0	0	0	0	1	0
1	1	1	1	1		1	1	1	1	1	1		1	1	1	1	1	1	0	0	0		1	1	1	1	1	1	0	0	0
0	0	0	0	0		0	0	0	0	0	0		0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	1	0	0

S_4

$\alpha = 1$

0
1
0
0
0
1
0
0
1
0
0
0

$\alpha = 2$

0	0	0
1	1	1
0	0	0
0	0	0
0	0	0
1	0	0
0	0	0
0	0	0
1	1	1
0	0	0
0	0	1
0	1	0

$\alpha = 3$

0	0	0	0	0	0	1
1	1	1	1	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
1	1	1	0	0	1	1
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	0	0	0	0

$\alpha = 4$

0	0	0	0	0	0	1	0	0	0
1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	0	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0
1	1	1	0	0	1	1	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	0
0	1	0	0	0	0	0	0	0	1

S_5

$\alpha = 1$

0
1
0
0
1
0
0
0
0
0
0
1

$\alpha = 2$

0	0	0	0	0	0	0
1	0	0	1	0	0	1
0	0	0	0	0	1	0
0	0	0	0	0	0	0
1	1	1	1	1	1	0
0	0	0	0	0	0	1
0	0	0	0	1	0	0
0	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
1	1	1	0	1	1	1

$\alpha = 3$

0	0	0	0	0	0	0	1
1	0	0	1	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	1
0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0
0	0	1	0	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	0	0	0
1	1	1	0	1	1	1	1

$\alpha = 4$

0	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	1	0	1	1
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	1	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1

S_6

$\alpha = 1$

0	0	0	0	0	1	0	1
0	0	0	1	0	0	1	0
1	1	0	0	1	0	0	0
1	0	1	1	0	1	0	0
0	0	0	0	1	0	0	0
0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0

$\alpha = 2$

0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	1	1	1	1
1	0	1	1	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1

$\alpha = 3$

0	0	0	0	0	1	0	1	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	1	1	1	1
1	0	1	1	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1

La asignación final resultó ser:

Horario final

Enfermera	Día 1	2	3	4
1	T	_	D	D
2	D	T	T	_
3	_	D	T	T
4	N	_	N	T
5	T	T	_	N
6	N	N	_	T

El costo de asignación total tuvo un valor de 6.

Aplicación 4.4.2

El objetivo de esta aplicación es mostrar el desarrollo de la función objetivo conforme el número máximo de columnas generadas en cada iteración de algunos problemas. En todos los casos se consideraron 5 iteraciones.

Problema	máximo de columnas por iteración	Iteración 1	2	3	4	5
1	5	2060	14	14	14	14
2	5	2052	6	6	6	6
3	5	56	6	6	6	6
4	5	71	37	21	6	6
	10	71	21	6	6	6
	15	71	21	6	6	6
	20	71	21	6	6	6
	25	71	21	6	6	6
5	5	4104	3081	2058	12	12
	10	4104	62	12	12	12
	15	4104	62	12	12	12

Conclusiones

La asignación de horarios a enfermeras es sólo una etapa de un proceso más extenso, el cual está denominado como planeación u organización de horarios para enfermeras. La mayoría de los autores consideran que este proceso se puede dividir en tres etapas: la de determinación de la cantidad de personal por categoría para cada turno y día de trabajo, la de asignación y por último la de reasignación. Siendo esta última la etapa en la que se hacen ajustes debido a inasistencia del personal y otro tipo de imprevistos. La metodología presentada en este trabajo constituye una propuesta para resolver el problema específico de asignación.

A pesar de que en los libros de texto pocas veces se hace referencia al problema de la organización de horarios, el problema de la asignación ha sido de interés para los científicos vinculados con la investigación de operaciones por más de 30 años. Específicamente asociado a las enfermeras por la complejidad de los problemas que se originan en el ambiente hospitalario. El primer intento reconocido para resolver este problema utilizando programación matemática fue el hecho por Michael Warner en 1976 y a partir de esta metodología han surgido infinidad de propuestas para resolver problemas de asignación de horarios. Entre ellas, las que utilizan métodos con heurísticas. El problema es que cada intento se ha hecho para resolver un problema en específico, lo que dificulta la tarea de comparar las metodologías propuestas.

Otra problemática en la implementación de las metodologías es que no hay un consenso en cuanto a lo que se considera como restricciones fuertes y restricciones suaves. Cada institución tiene sus criterios particulares en cuanto a la clasificación de las restricciones o por ejemplo en cuanto a lo que se considera un patrón de trabajo indeseable. Esto ocasiona en consecuencia que muchos de los algoritmos propuestos para resolver problemas de asignación de horarios no puedan ser probados en distintos ambientes hospitalarios.

Muchos de los métodos que se han propuesto a lo largo del tiempo no han sido puestos en práctica y de hecho, algunos de ellos ni siquiera probados con datos reales. Considero que este punto es de gran relevancia, ya que el probar los métodos con información real puede ser un primer paso para determinar la eficiencia de las metodologías propuestas.

La metodología presentada en este trabajo de tesis constituye una herramienta de gran ayuda para las instituciones hospitalarias y de beneficio también para el personal de enfermería.

La metodología permite la obtención de una buena asignación de horarios, ya que en el modelo se considera un criterio de evaluación de gran relevancia que es la aversión de los empleados por ciertos patrones de trabajo y sus preferencias por ciertos turnos a laborar.

Un punto importante es que el modelo y la metodología presentados no limitan su aplicación al sector hospitalario, ya que fueron diseñados para permitir adaptarse con facilidad a las prioridades y a las necesidades según el ambiente laboral en el que se desee utilizarlo.

El procedimiento propuesto constituye una herramienta que favorece la asignación de horarios. Contribuye eficientemente a que las organizaciones hospitalarias consigan distribuir al personal en las diferentes tareas tomando en cuenta las preferencias del personal de enfermería. La asignación se hace conforme a las necesidades del hospital y las características del personal del que dispone; lo cual es de gran utilidad en la búsqueda de la permanencia del personal.

En cuanto a los resultados obtenidos a partir de la implementación de la metodología en MATLAB:

- Con las aplicaciones propuestas, se observó que la metodología presentada es eficiente para resolver los problemas de asignación de horarios, principalmente por su contribución en la generación de opciones factibles para cada enfermera. Se pudo observar la afectación en los resultados dependiendo de qué tantas enfermeras hagan solicitudes particulares en cuanto a sus horarios y qué tanto se contrapongan unas con otras.

- La experimentación con el programa permitió comprobar que el tiempo de ejecución no sólo está relacionado con la cantidad de variables y de restricciones (tamaño del problema), sino también con los datos de partida del caso que se pretende resolver.

- La metodología presentada es una propuesta eficiente para la asignación de horarios a enfermeras si se considera la forma en la que se plantea el problema de asignación y en cómo se generan los horarios factibles. En cuanto a la implementación, la falta de una propuesta para generalizar la metodología en lo que concierne al tratamiento de restricciones fuertes y débiles para poder ser aplicada a cualquier ambiente hospitalario representa una desventaja, ya que cada vez que se desee cambiar de ambiente hospitalario se debe de implementar una forma distinta de evaluación de factibilidad de los horarios generados.

Apéndice A

Branch and Bound

El método “Branch and Bound” es uno de los métodos denominados de enumeración implícita para encontrar la solución óptima de problemas. Este método es muy utilizado en la práctica para resolver tanto problemas de programación entera como de optimización combinatorial, sobre todo porque este tipo de problemas tienen un número de soluciones factibles muy grande. La enumeración explícita sería imposible debido al crecimiento exponencial de éstas soluciones. Por lo que éste método ha resultado ser muy eficiente bajo estas circunstancias.

Originalmente se define al método “Branch and Bound” como un método de búsqueda exhaustiva a través de un árbol de búsqueda. La razón principal por la que se ha tenido que recurrir a estos métodos para resolver problemas de programación entera es debido a que no se tienen condiciones de optimalidad para verificar si una solución factible encontrada en éstos problemas es óptima o no. Por ejemplo, en programación lineal, cuando se tiene una solución candidata, se checa si existe una dirección factible de mejora para moverse, si no existe, entonces la solución candidata es óptima, de lo contrario se puede encontrar una dirección hacia la cual moverse, lo que proporcionará una mejor solución, verificándose así que la solución candidata no era óptima. Este tipo de condiciones de optimalidad global no existen en programación entera ni en optimización combinatorial, por lo que para verificar la optimalidad de una solución factible es necesario comparar ésta con todas las otras posibles soluciones Benli(1999).

Debido a las condiciones de los problemas enteros y de optimización combinatorial, esto resulta imposible de realizar por lo que es necesario un método que implícitamente lleve a cabo ésta acción, es decir, haciendo una enumeración explícita “parcial” de las soluciones factibles.

Como ejemplo tomaremos el caso hipotético de la asignación de dos empleados a cuatro días de trabajo. Los dos empleados pueden ser asignados a cada día. Se supondrá que el costo de la asignación considerando las preferencias de las empleados ya se ha determinado para cada empleado en cada día.

Esto nos lleva a enumerar las soluciones del siguiente problema de programación entera.

El problema consiste en asignar días a laborar a dos empleados de una empresa. Se define la variable y_{ij} igual a 1 si el día *j* se le asigna al empleado *i* y 0 de no ser asignado. Lo que se busca en el problema es minimizar el costo de la asignación de los días a laborar. Para el ejemplo se considerarán 4 días.

Si se tuvieran que enumerar las asignaciones, al ser un problema en donde las variables de decisión son binarias, el número de posibles asignaciones es $2^4 = 16$, ya que el número de

días es cuatro, por lo que el árbol de enumeración asociado a éste problema es el mostrado por la figura 1.

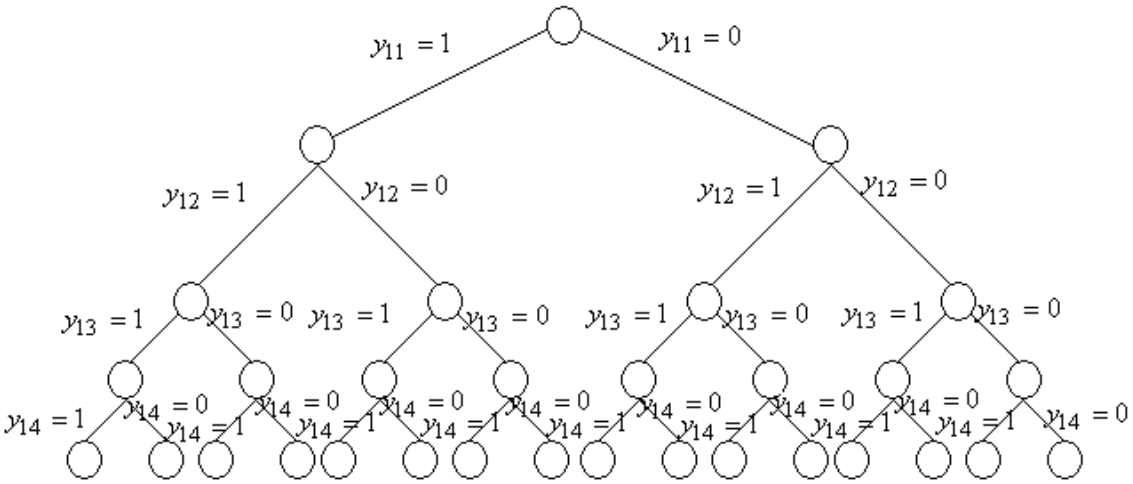


Fig. 1 Árbol de enumeración

El primer nodo el cual es llamado nodo raíz y es desde donde se comienza, es decir, cuando aún no se ha hecho ninguna asignación a las enfermeras. El nodo raíz se ramifica en dos nodos, los cuales son sus descendientes. El primer nodo corresponde a asignarle al empleado 1 el día 1 y el segundo nodo corresponde a hacer la misma asignación al empleado 2. Cada nodo corresponde a una asignación parcial.

Los nodos descendientes del nodo 1 implican asignarle el turno 1 a la enfermera 1 y la posibilidad de asignarle también el turno dos a ésta enfermera o el asignárselo a la enfermera dos. De igual forma cada nodo del árbol corresponde a una asignación. Sólo los últimos nodos corresponden a un horario completo. Por lo que el árbol de enumeración genera todas las posibles soluciones al problema.

En éste caso resulta fácil enumerar explícitamente las posibles soluciones del problema, pero como ya se mencionó, conforme crece el número de variables, el número de posibles soluciones crece exponencialmente, lo que hace imposible hacerlo en problemas reales, ya que éstos generalmente involucran muchas variables. De ésta manera el buscar la solución a éste tipo de problemas hace necesario el uso de un método que no haga una búsqueda exhaustiva en el árbol completo.

El método “Branch and Bound, consiste en que al tomar un nodo y determinar que en sus descendientes no se va a obtener una solución óptima, entonces ese nodo es “podado”, es decir, que ya no se consideran para una posible solución. De ésta forma al ir eliminando nodos y sus descendientes, el problema se va haciendo más fácil de manejar. Por medio del uso de cotas para la función objetivo que se quiere optimizar y con el valor de la “mejor”

solución al momento, el algoritmo “Branch and Bound” es capaz de buscar en partes del espacio de soluciones implícitamente.

En el algoritmo, el nodo raíz se considera como el problema original que se quiere resolver y cada descendiente un subproblema de éste, de ésta forma el método consiste en la partición iterativa del conjunto de soluciones factibles para generar subproblemas del problema original. La figura 2 ejemplifica el espacio de búsqueda del algoritmo “Branch and Bound”.

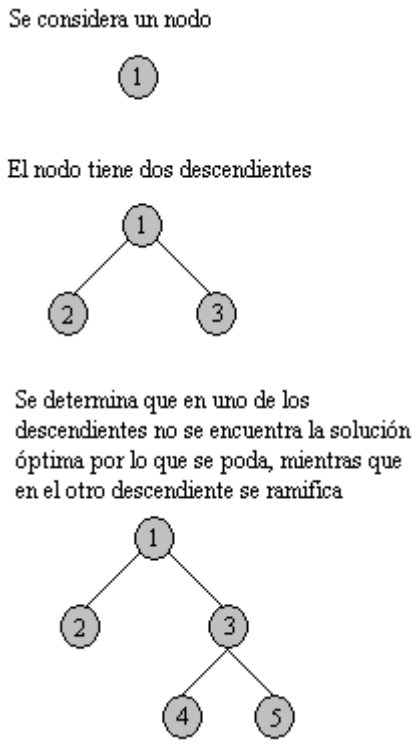


Fig. 2 Espacio de búsqueda del algoritmo “Branch and Bound”

Cada iteración del algoritmo involucra tres procesos: seleccionar el nodo que será inspeccionado, el cálculo de una cota y la ramificación, la cual es la división del espacio de soluciones en dos o más subespacios, los cuales serán analizados en posteriores iteraciones (Clausen1999). Existen muchas formas de llevar a cabo estos procesos, pero ninguna combinación particular de estos procesos resulta ser eficiente para todos los casos, por lo que se tienen que tomar en cuenta las particularidades del problema para poder elegir el método adecuado para cada uno de los tres procesos mencionados. De ésta forma se tiene que diseñar un algoritmo “Branch and Bound” específico para cada clase de problemas.

Siguiendo con el método cada subproblema que se genera durante el proceso del algoritmo se resuelve para encontrar una cota. Si por ejemplo hablamos de un problema de maximización, y el valor de una función objetivo de un subproblema dado tiene una cota

superior menor al valor de la función objetivo en una solución factible entera conocida, el cual es denominado como “incumbente”, entonces la solución óptima del problema entero original no se encuentra en el conjunto de soluciones factibles relacionadas a dicho subproblema. De ésta forma las cotas superiores de éstos subproblemas se van usando para garantizar la optimalidad de la solución sin tener que hacer la búsqueda explícita de ésta.
La búsqueda de la solución termina cuando ya se han inspeccionado todos los nodos con los que se cuenta y la mejor solución sigue siendo la incumbente.

Si seguimos en el caso de las enfermeras, se puede ejemplificar éste proceso, primero una manera de encontrar una solución “incumbente” es tomando un caso extremo que fuera asignarle a una sola enfermera todos los turnos.
Supongamos que esta solución da como resultado un valor de la función objetivo igual a doce, la cual se considera como la peor solución. De ésta forma se descartaran todas las soluciones que den como resultado un valor de la función objetivo mayor que éste valor “incumbente” encontrado, ya que lo que se busca en este caso es minimizar la función.

Una manera de encontrar una cota en determinado nodo es resolviendo una relajación de un problema. Por ejemplo teniéndose el problema original una manera de relajarlo es quitando la restricción concerniente a que las variables tomen valores enteros. Al resolver el problema de programación lineal se obtiene una cota.
La forma en la que se van obteniendo las cotas es muy importante ya que conforme sean más estrechas, el árbol se podrá “podar” más, reduciendo el número de alternativas a explorar explícitamente.

Un aplicación del método “Branch and Bound”, para resolver un problema de programación entera se mostrará con un ejemplo muy sencillo.

Supongamos que se desea resolver el siguiente problema de programación entera:

$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1, x_2 \in Z^+
\end{aligned}
$$

La región de soluciones factibles asociada al problema se ilustra en la figura 3.

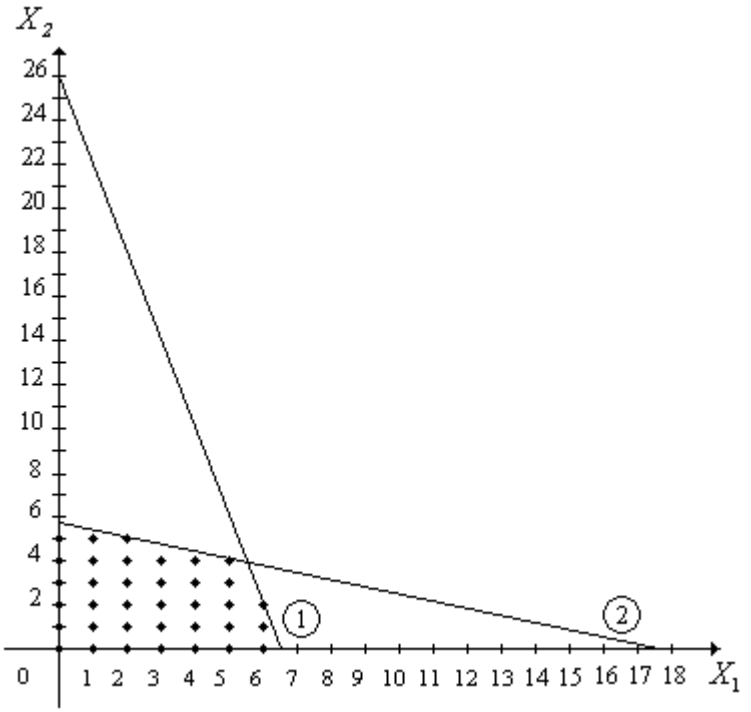


Fig.3 Región de soluciones factibles del problema

Al inicio se establece el valor del incumbente como $\underline{z} = -\infty$.
Relajando el problema se obtiene la región de soluciones mostrada en la figura 4.

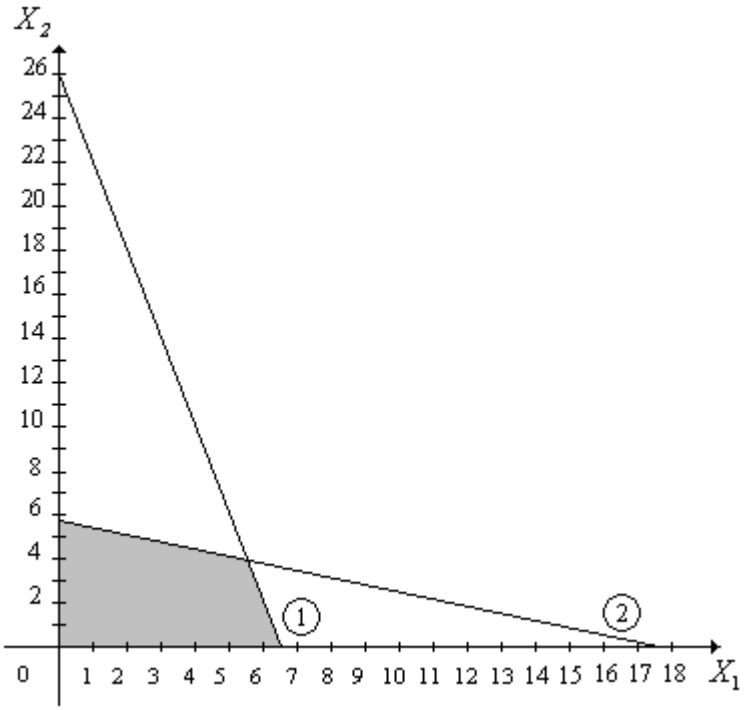


Fig. 4 Región de soluciones factibles al relajar el problema original

Al aplicar el método simplex al problema relajado se obtiene la siguiente solución: $x_1 = 5.5, x_2 = 4$ y z = 48.5. Como no se obtuvo una solución entera el valor del incumbente permanece igual. Como el valor que se obtuvo para x_1 es fraccional, se elige esta variable para ramificar. Los problemas que se generan a partir de la ramificación son:

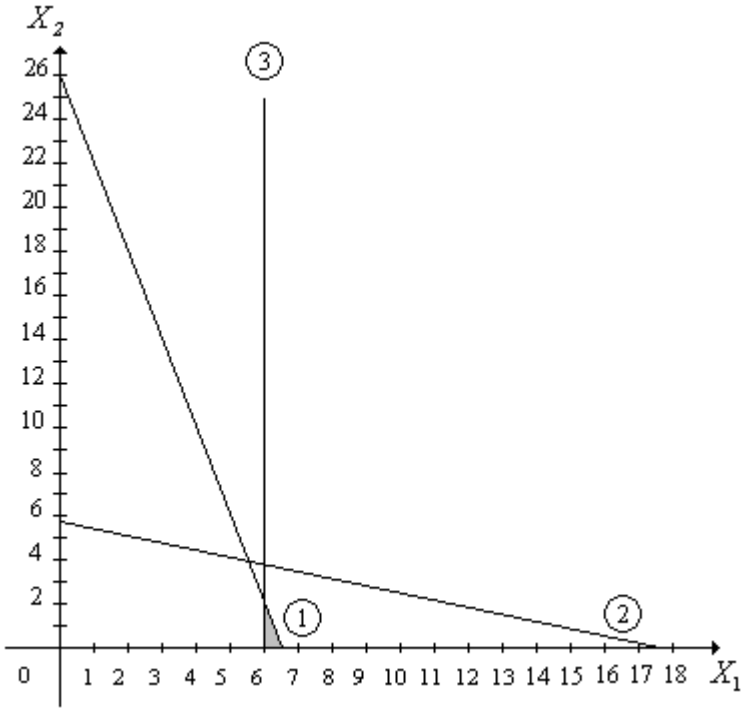
Problema 1

$$\begin{aligned} Max\ & 3x_1 + 8x_2 \\ s.a.\ & 4x_1 + x_2 \le 26 \\ & 2x_1 + 6x_2 \le 35 \\ & x_1 \ge 6 \\ & x_1, x_2 \ge 0 \end{aligned}$$

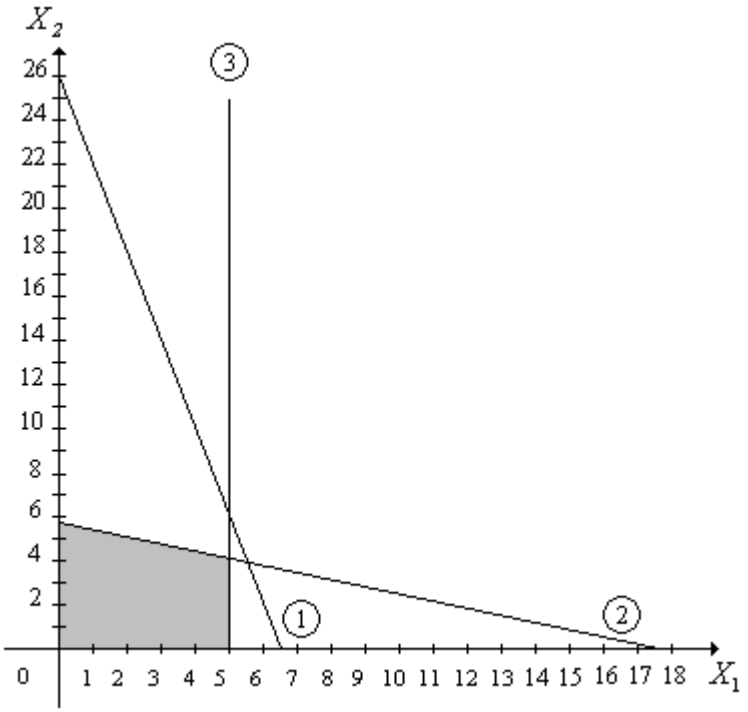
Problema 2

$$\begin{aligned} Max\ & 3x_1 + 8x_2 \\ s.a.\ & 4x_1 + x_2 \le 26 \\ & 2x_1 + 6x_2 \le 35 \\ & x_1 \le 5 \\ & x_1, x_2 \ge 0 \end{aligned}$$

Las regiones de soluciones factibles asociadas a los problemas son las ilustradas en la figura 5.



Región de soluciones factibles problema 1



Región de soluciones factibles problema 2
Fig. 5

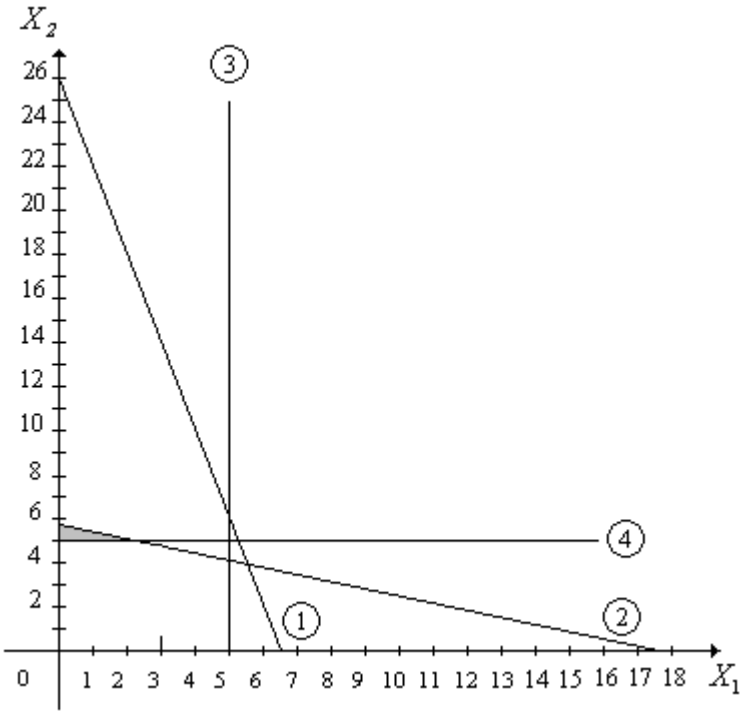
Al resolver el problema 1 se obtiene la siguiente solución $x_1 = 6, x_2 = 2$ y z = 34. Al obtener una solución entera cambiamos el valor del incumbente a $\underline{z}$ =34. Al resolver el problema 2 se obtiene la siguiente solución: $x_1 = 5, x_2 = 4.166$ y z = 48.33, como la variable x_2 es fraccional, esto nos da pie a generar dos problemas más.

Problema 2.1

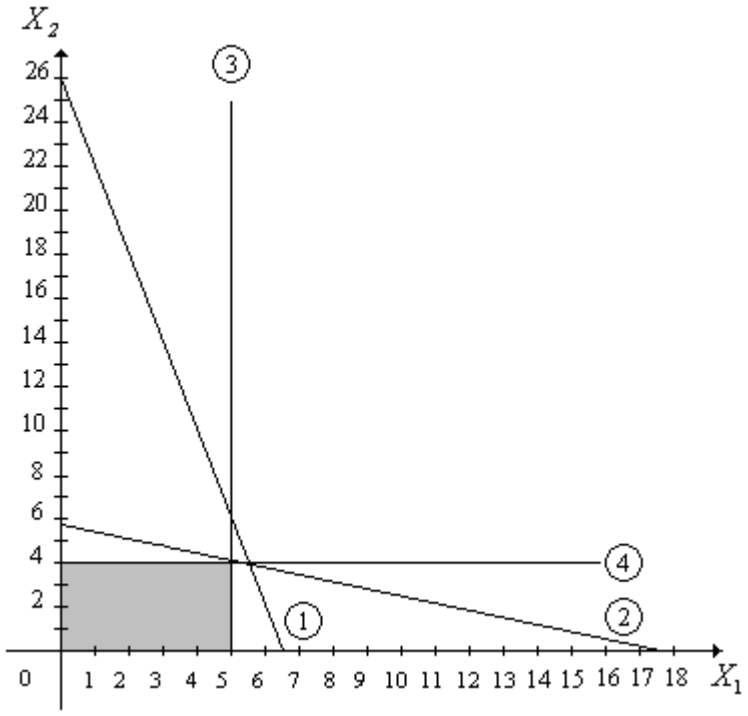
$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \ge 5 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

Problema 2.2

$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \le 4 \\
& x_1, x_2 \ge 0
\end{aligned}
$$



Región de soluciones factibles del problema 2.1



Región de soluciones factibles del problema 2.2

Fig. 6

Ahora al resolver el problema 2.1 se obtiene la siguiente solución $x_1 = 2.5, x_2 = 5$ y z = 47.5. Esto genera los problemas 2.1.1 y 2.1.2 que se resolverán posteriormente. Al resolver el problema 2.2 se obtiene la siguiente solución: $x_1 = 5, x_2 = 4$ y z = 47, como este valor de z es mayor que el del incumbente, se actualiza éste valor $\underline{z} = 47$.

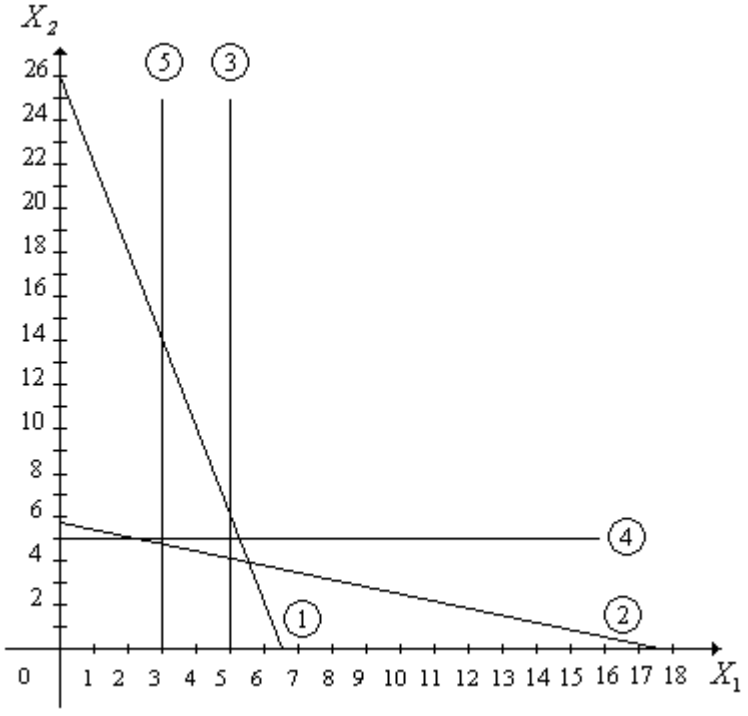
Los problemas que se generaron a partir del problema 2.1 son:

Problema 2.1.1

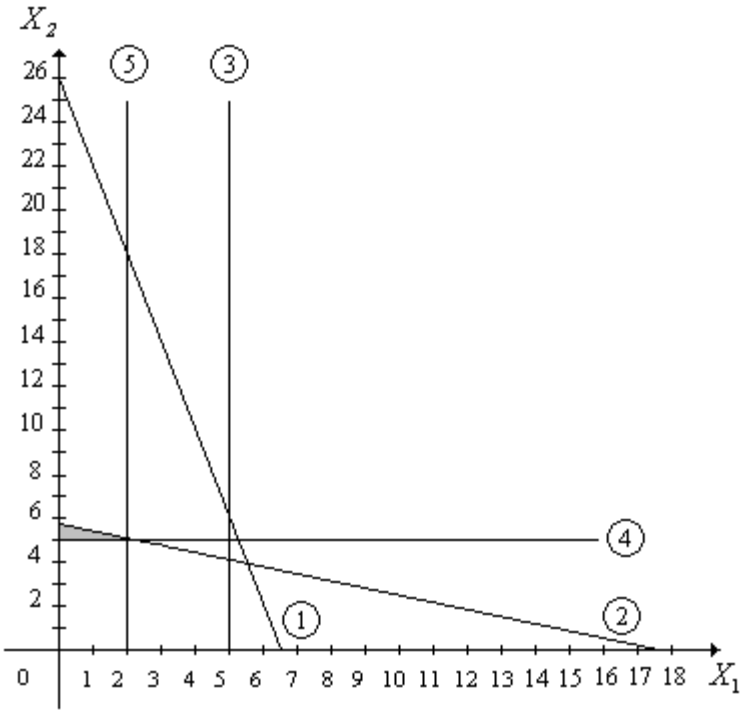
$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \ge 5 \\
& x_1 \ge 3 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

Problema 2.1.2

$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \ge 5 \\
& x_1 \le 2 \\
& x_1, x_2 \ge 0
\end{aligned}
$$



Región de soluciones factibles del problema 2.1.1



Región de soluciones factibles del problema 2.1.2

Fig. 7

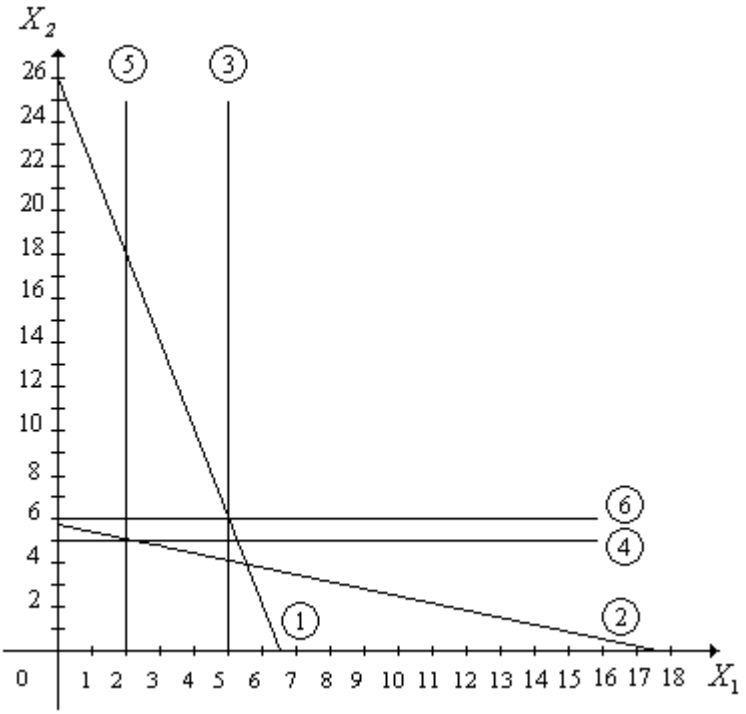
Al resolver ambos problemas se obtiene, para el problema 2.1.1 no existe solución factible. Mientras que al resolver el problema 2.1.2 se obtiene la solución $x_1 = 2, x_2 = 5.16$ y z = 47.33, lo que nuevamente da origen a dos problemas más:

Problema 2.1.2.1

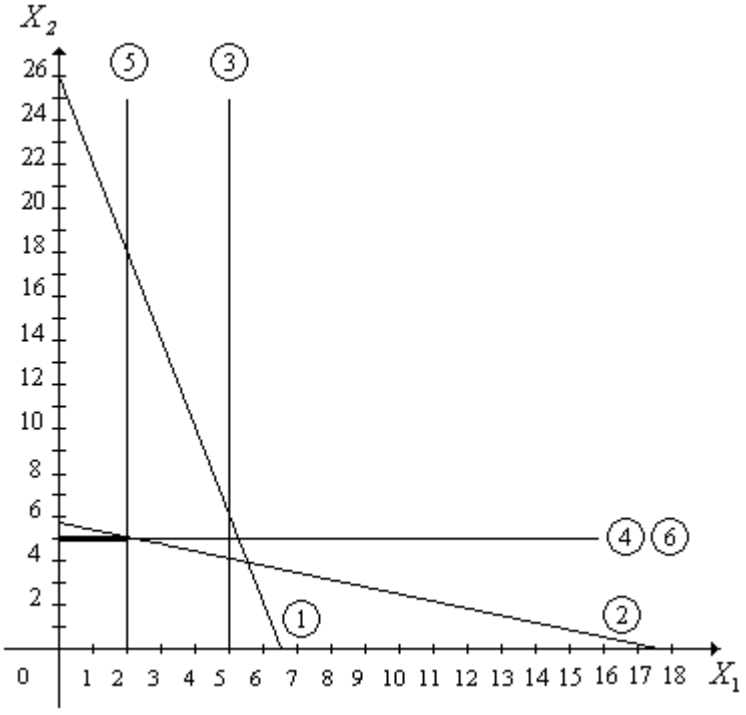
$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \ge 5 \\
& x_1 \le 2 \\
& x_2 \ge 6 \\
& x_1, x_2 \ge 0
\end{aligned}
$$

Problema 2.1.2.2

$$
\begin{aligned}
Max\ & 3x_1 + 8x_2 \\
s.a.\ & 4x_1 + 2x_2 \le 26 \\
& 2x_1 + 6x_2 \le 35 \\
& x_1 \le 5 \\
& x_2 \ge 5 \\
& x_1 \le 2 \\
& x_2 \le 5 \\
& x_1, x_2 \ge 0
\end{aligned}
$$



Región de soluciones factibles del problema 2.1.2.1



Región de soluciones factibles del problema 2.1.2.2

Fig. 8

Para el problema 2.1.2.1 no existe solución factible, mientras que para el problema 2.1.2.2 la solución que se encontró es: $x_1 = 2, x_2 = 5$ y z = 46. A pesar de que la solución es entera el incumbente no se modifica ya que el valor obtenido es menor a él.

Como estos problemas no pueden seguir siendo ramificados para obtener una solución entera mejor, entonces el valor del incumbente es óptimo. Por lo que el resultado del problema original es:

$$x_1 = 5$$
$$x_2 = 4$$
$$z = 47$$

El árbol final que resulta al resolver el problema utilizando el método de "Branch and Bound" es el que se muestra en la figura 9.

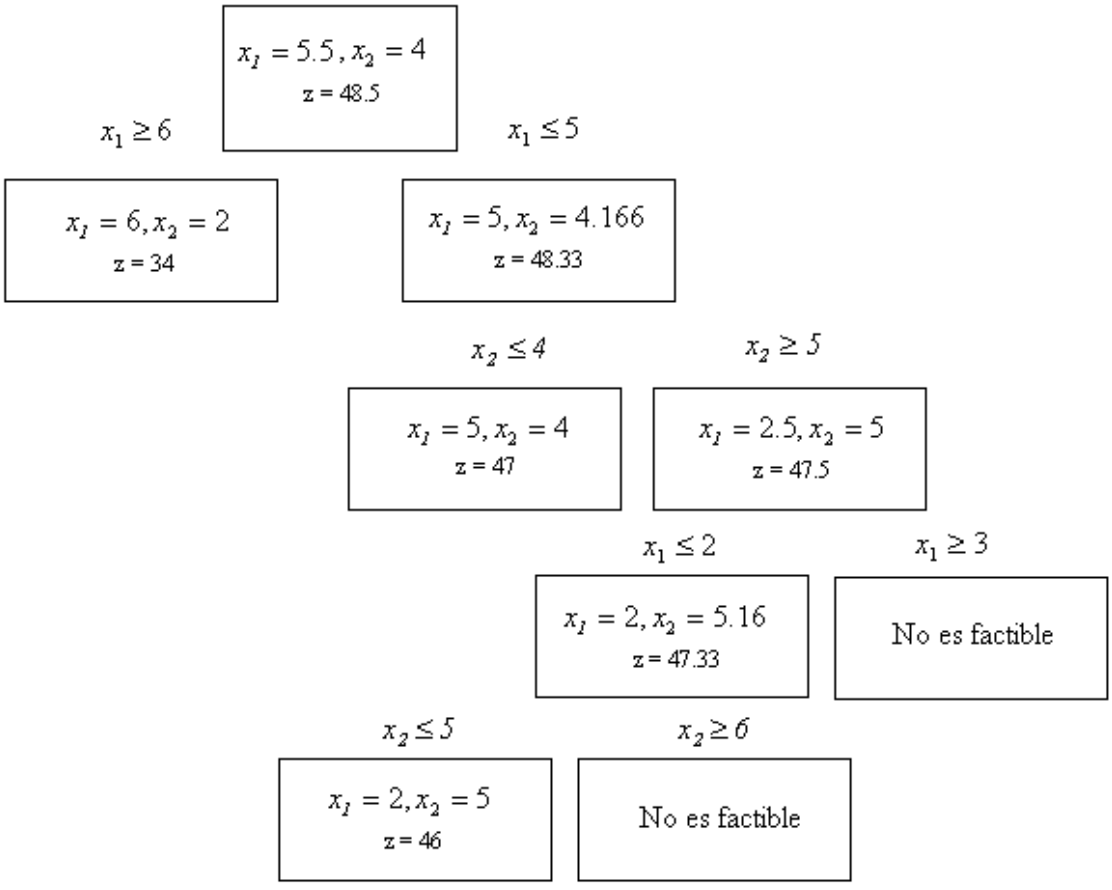


Fig. 9 Estructura del árbol final

Vecindad de desplazamiento y vecindad de intercambio [3]

Dos vecindades clásicas utilizadas en los algoritmos de búsqueda local son la vecindad de desplazamiento (Shift Neighborhood) y la vecindad de intercambio (Swap Neighborhood):

Para poder definir mejor éstas vecindades tomaremos como ejemplo un problema de satisfacción de restricciones en donde se tiene un conjunto de variables $\{X_1, X_2, X_3, X_4\}$, cuyo dominio por simplicidad es el mismo para todas ellas y está dado por el conjunto $D=\{1,2,3\}$. Se definirá una variable X_{ij} de la siguiente manera:

$$X_{ij} = \begin{cases} 1 & si \;\; X_i = j, \; con \; j \in D \\ 0 & e.o.c. \end{cases}$$

Usando estas variables se representará una asignación con un vector *x* cuyas componentes serán ceros y unos. Así, por ejemplo la asignación $\{X_1 \leftarrow 1, X_2 \leftarrow 2, X_3 \leftarrow 2, X_4 \leftarrow 3\}$, será representada por el vector:

$$x = \{100|010|010|001\}$$

Se asume que $\sum_{j \in D} X_{ij} = 1$, para i=1,2,3,4.

Vecindad de desplazamiento (Shift neighborhood)

Se define la vecindad *N(x)* de la siguiente forma: Sea $x(X_{ij} \leftarrow 1)$ la solución obtenida de *x* al intercambiar X_{ij} de 0 a 1. Esto implica que se haga el cambio $X_{ij'} \leftarrow 0$, para la $X_{ij'}$ que satisfacía $X_{ij'}$=1. Así, la vecindad se define por:

$$N(x) = \{x(x_{ij} \leftarrow 1) \mid x_{ij} = 0, \; i = 1,...,4, \; j \in D\}.$$

Así, una vecindad de desplazamiento (shift neighborhood), se genera como su nombre lo dice al desplazar un 1 en la solución actual hacia la posición en donde halla un 0.

Por medio de un movimiento de desplazamiento a partir de la solución actual:

[3] Nonobe, K. e Ibaraki, T. (1998). A tabu search approach to the constraint satisfaction problem as a general problem solver. European Journal of Operational Research 106, 599–623.

$$x = \{100|010|010|001\}$$

Se obtiene la solución vecina:

$$x' = \{100|010|010|100\}$$

Vecindad de intercambio (Swap neighborhood)

Esta vecindad considera problemas que tiene la restricción:

$$\sum_{X_i \in V'} X_{ij} = a_j, \quad j \in D$$

En donde *V'* es un subconjunto del conjunto de variables *V* y a_j es una constante, generalmente igual a 1. Suponga que *x* es una solución que cumple con dicha restricción y que satisface que $X_{i_1 j_1} = 1$ y $X_{i_2 j_2} = 1$, con $i_1 \neq i_2$ y $j_1 \neq j_2$. Entonces al intercambiar $X_{i_1 j_1}$ y $X_{i_2 j_2}$, es decir si ($X_{i_1 j_2} \leftarrow 1$) y ($X_{i_2 j_1} \leftarrow 1$) obtenemos una solución vecina.

De ésta forma la vecindad $N_s(x)$ se define:

$$N_s(x) = \left\{ x(x_{i_1 j_2} \leftarrow 1, x_{i_2 j_1} \leftarrow 1) \middle| \begin{array}{l} x_{i_1 j_2} = 1, x_{i_2 j_1} = 1, \\ i_1 \neq i_2, j_1 \neq j_2, \ i_1, i_2 \in V', j_1, j_2 \in D \end{array} \right\}$$

Así, una vecindad de intercambio (swap neighborhood), se genera al intercambiar dos asignaciones que no pueden ser las mismas.

Por medio de un movimiento de intercambio de la solución actual:

$$x = \{100|010|010|001\}$$

Se obtiene la solución vecina:

$$x' = \{010|100|010|001\}.$$

Apéndice B

Códigos fuente de programa de asignación

Programa de asignación

```
%Brenda B. Carrasco Enríquez
%optimalidad

function []=sched2p()

clc;
a=[];
a1=[];

n=input('Ingrese número de enfermeras de planta\n');

for i=1:n
    HENF=['a1(',int2str(i),',:)=input(''Ingrese horario sugerido por la
enfermera' int2str(i),'\n'');'];
    eval(HENF)
end;

for i=1:n
    HENF=['a(i,:)=a1(',int2str(i),',:);'];
    eval(HENF)
end;

a=char(a);

[n d]=size(a);
pMIN=size(d,1);
pMAX=size(d,1);
z=zeros(3*d,1);
qMIN=zeros(3*d,1);
qMAX=zeros(3*d,1);
dayoff=size(n,1);
h=zeros(n,3*d);
costo=zeros(n,1);
costo1=zeros(n,1);
[fs,d1]=aleatorios(d);
viol=[];
alfa=1;
bandera=0;

beta=.85;

%Demandas de personal por turno
for i=1:d
   if i==1
       pMIN(i,1)=input('Ingrese demanda MIN por Turno de la mañana\n');
   elseif i==2
       pMIN(i,1)=input('Ingrese demanda MIN por Turno de la tarde\n');
   elseif i==3
       pMIN(i,1)=input('Ingrese demanda MIN por Turno de la noche\n');
   end;
end;
```

```
for i=1:d
   if i==1
       pMAX(i,1)=input('Ingrese demanda MAX por Turno de la mañana\n');
   elseif i==2
       pMAX(i,1)=input('Ingrese demanda MAX por Turno de la tarde\n');
   elseif i==3
       pMAX(i,1)=input('Ingrese demanda MAX por Turno de la noche\n');
   end;
end;

%Restricciones del hospital
ws=input('Ingrese número máx de días consecutivos a laborar\n');
c=input('Ingrese número mínimo de días a trabajar por enfermera\n');
dl=input('Ingrese número de días libres por enfermera\n');

fprintf('\nSolicitudes individuales de las enfermeras\n');

%Requerimientos individuales
for i=1:n
   fprintf('\nIndique día libre requerido por la enfermera %d\n',i);
   fprintf('Teclee 0 si la enfermera no desea hacer solicitudes\n');
   dayoff(i,1)=input('');
end;

pref=[];
for i=1:n
    HENF=['fprintf(''Indique grado de aversión de la enfermera
',int2str(i),' por los siguientes patrones de
trabajo:\n'');pref(',int2str(i),',1)=input('' 000 100
000\n'');pref(',int2str(i),',2)=input('' 000 010
000\n'');pref(',int2str(i),',3)=input('' 000 001
000\n'');pref(',int2str(i),',5)=input('' 100 001 \n'');fprintf('' Días de
trabajo consecutivos igual a %d
\n'',ws);pref(',int2str(i),',4)=input('''');'];
    eval(HENF)
end;

%Definición de parámetros para controlar el tamaño del problema
maxcol=input('Ingrese número máximo de columnas por enfermera\n');
vmax=input('Ingrese número máximo de violaciones por enfermera\n');
maxcoltot=input('Ingrese número máximo de columnas total\n');
alfamax=input('Ingrese número máximo de iteraciones\n');
viol=input('Ingrese número de violaciones de los horarios del periodo de
planeación anterior\n');

%crea vectores de demanda para todo el horizonte de planeación

%demanda mínima
for i=1:3*d
    s=mod(i,3);
    if s==1
        qMIN(i,1)=pMIN(1,1);
    elseif s==2;
        qMIN(i,1)=pMIN(2,1);
    elseif s==0
        qMIN(i,1)=pMIN(3,1);
```

```
    end;
end;

%demanda máxima
 for i=1:3*d
     s=mod(i,3);
     if s==1
      qMAX(i,1)=pMAX(1,1);
   elseif s==2;
      qMAX(i,1)=pMAX(2,1);
   elseif s==0
      qMAX(i,1)=pMAX(3,1);
   end;
   end;


%Crea matriz de periodos 0-1

for i=1:n
   k=1;
   for j=1:d
      while k<(3*d)
         if a(i,j)==' '
            h(i,k)=0;
            k=k+3;
            break;
      elseif a(i,j)=='D'
         h(i,k)=1;
         k=k+3;
      break;
      elseif a(i,j)=='T'
         h(i,k+1)=1;
         k=k+3;
      break;
      elseif a(i,j)=='N'
         h(i,k+2)=1;
         k=k+3;
      break;
      end;
      end;
   end;
   end;

%Cuantifica violaciones de todos los horarios y verifica si el horario
completo es óptimo.
maxvio=maxviol(viol);

[o viol1]=optimo2(a,h,ws,dayoff,c,qMIN,qMAX,pref);

if o==1
    hbase=h';

%Horarios
[i j]=size(hbase);

%Se calcula el costo de los horarios suministrados
for k=1:j
```

```
    costo(k,1)=2^(viol1(k));
end;

costo
pause

for k=1:j
 if costo(k,1)==1;
 costo1(k,1)=.5;
 else
     costo1(k,1)=costo(k,1);
 end;
end;

costo=costo1;

costo
pause
%Se crea una matriz de horarios para cada enfermera
for n = 1:j
      HENF= ['A',int2str(n),' =hbase(:,n);
costo',int2str(n),'=[costo(n,1)];'];
      eval(HENF)
end;

%Se crea matriz que almacenará los identificadores de las columnas de
cada
%enfermera
for n = 1:j
    HENF=['htabenf',int2str(n),'=zeros(1,50);'];
    eval(HENF)
end;

%A cada horario se le asigna un identificador
for n=1:j
  HENF=[' [ar htabenf',int2str(n),'
cos]=redun2(A',int2str(n),',htabenf',int2str(n),',costo',int2str(n),',fs,
d1);'];
  eval(HENF)
end;


while alfa<=alfamax

%Se crea variable tc total de columnas por cada enfermera
    for n = 1:j
        HENF= ['[r,s]=size(A',int2str(n),'); tc(n,1)=s;'];
        eval(HENF)
    end;

%Se crean columnas nuevas para cada enfermera


for n=1:j
    HENF=['[ncol',int2str(n),' c',int2str(n),'
htabenf',int2str(n),']=gencol2(hbase,hbase(:,n),c,ws,qMIN,qMAX,dayoff(',i
```

```
nt2str(n),'),dl,maxvio(',int2str(n),'),maxcol,tc,tc(n,1),fs,d1,htabenf',i
nt2str(n),',pref(',int2str(n),',:));'];
    eval(HENF)
end;

    k=1;
    ind=[];
    for n=1:j
        HENF=['if isempty(ncol',int2str(n),')==1 ind(k)=n;k=k+1;end;'];
        eval(HENF)
    end;

 %%Se agregan columnas nuevas al conjunto de horarios de cada enfermera

for n=1:j
    if isempty(find(n==ind))==1
        HENF=['A',int2str(n),'=[A',int2str(n), ' ncol',int2str(n),'];
costo',int2str(n),'=[costo',int2str(n),';c',int2str(n),'];'];
        eval(HENF)
    elseif isempty(find(n==ind))==0
    end;
end;

%Se juntan todas las matrices en una sola matriz para crear la estructura
del modelo

    A=[];
    costo=[];
    for n = 1:j
       HENF=['A=[A A',int2str(n),'];costo=[costo;costo',int2str(n),'];'];
        eval(HENF)
    end;


    [i0 j0]=size(A);

   %Se crean restricciones para cada enfermera

    for n = 1:j
        HENF=['z',int2str(n),'=zeros(1,j0);'];
        eval(HENF)
    end;

    x=0;
   for n = 1:j
       HENF=['[p q]=size(A',int2str(n),'); for k=1:q
z',int2str(n),'(1,k+x)=1;end;'];
       eval(HENF)
       x=x+q;
   end;


%se crea matriz de todas las restricciones
   rest=[];
   for n = 1:j
     HENF=['rest=[rest;z',int2str(n),'];'];
     eval(HENF)
```

```
   end;

   qrest=[ones(j,1)];


   [h f exitflag bandera
costos]=opbin(A,rest,qMIN,qMAX,qrest,costo,bandera);

%*************************************************************
   if (exitflag==2)&(j0>maxcoltot)
    alfa=alfamax+1;
   elseif (exitflag==2)&(j0<=maxcoltot)
    alfa=alfa+1;
   elseif (exitflag==1)&(j0<=maxcoltot)
     if bandera==0
     hbase=h;
     alfa=alfa+1;
     else
     hbase=h;
     alfa=alfa+1;
     end;
   elseif (exitflag==1)&(j0>maxcoltot)
     alfa=alfamax+1;
     break;
   end;

%*************************************************************
end; %-----end while
    elseif o==0;
        disp('El horario es óptimo');
    return;
end;


%Determina qué turno trabaja la enfermera el día j

hbase=hbase';
[n m]=size(hbase);

for i=1:n
   j=1;
   m1=1;
   while j< m%-1
          if  (hbase(i,j)==1)
             HOR(i,m1)='D';
             m1=m1+1;
          elseif (hbase(i,j+1)==1)
             HOR(i,m1)='T';
             m1=m1+1;
          elseif (hbase(i,j+2)==1)
             HOR(i,m1)='N';
             m1=m1+1;

          elseif (hbase(i,j)==0)&(hbase(i,j+1)==0)&(hbase(i,j+2)==0)
             HOR(i,m1)=' ';
             m1=m1+1;
           end;
```

```
          j=j+3;
   end;
end;

    disp('Horarios');
    disp(hbase');
 [v1 v2]=size(HOR);

 for n = 1:v1
     HENF=['disp(''Horario enfermera ',int2str(n),':''
);disp(HOR(',int2str(n),',:));'];
     eval(HENF)
   end;


   fprintf('\nCosto total de la asignación:%d\n',f);

  for n = 1:v1
     HENF=['disp(''Costo de asignación de horario de la enfermera
',int2str(n),':'' );disp(costos(n));'];
     eval(HENF)
   end;
```

Programa que genera números aleatorios

```
function [fs,d]=aleatorios(j);
clc; format long;

 d=[];
 fs=[];

for i=1:4
 fs(i)=ceil(20*rand(1));
end;


for n = 1:j
      HENF= ['d',int2str(n),'=ceil(30*rand(1));'];
      eval(HENF)
            end;

  for n = 1:j
      HENF= ['d=[d d',int2str(n),'];'];
      eval(HENF)
            end;
```

Programa que genera intercambios tomando en cuenta demandas de personal

```
function [a2,intercambio]=clasprom3(h,henf,qMIN,qMAX)
clc; format long;


[m n]=size(h);

a2=[];
```

```
intercambio=[];
A=zeros(m,2);
B=zeros(m,2);
A1=[];
B1=[];

% Crea conjunto de índices de periodos en los que se puede disminuir el
número
%  de enfermeras registradas y la enfermera está asignada para
trabajarlos

for i=1:m
   k=1;
   if henf(i,1)==1 &&(sum(h(i,:))>qMIN(i,1))
      A(i,k)=i;
      A(i,k+1)=1;
      k=k+2;
   else
      k=k+2;
   end;
end;

% Crea conjunto de índices de periodos en los que se puede aumentar el
número de
%  enfermeras registradas y la enfermera no está asignada para
trabajarlos

for i=1:m
   k=1;
   if (henf(i,1)==0)&&(sum(h(i,:))<qMAX(i,1))
      B(i,k)=i;
      B(i,k+1)=1;
      k=k+2;
   else
      k=k+2;
   end;
end;

[i1 j1]= find(A(:,1));
A=[i1 j1];
[i2 j2]= find(B(:,1));
B=[i2 j2];
[pa qa]=size(A);
[pb qb]=size(B);

if isempty(A)==0 && isempty(B)==0


   a2=ones(m,pb);

   for i=1:m
        a2(i,:)=henf(i,1)*a2(i,:);
    end;



    U=1;
```

```
sin=1;
%for sin=1:pa*pb
while sin<=(pb*pa)
for K=1:pb
        for K1=1:pa
            a2(B(K,1),sin)=1;
            a2(A(K1,1),sin)=0;
            intercambio(U,:)=[A(K1,1) B(K,1)];
          U=U+1;
          sin=sin+1;
    %        break;
      end;
     %break;
  end;
break;
end;



    % for K=1:pb
  %    for K1=1:pa
 %a2(B(K,1),K)=1;
 %a2(A(K1,1),K)=0;
 %intercambio(K,:)=[A(K1,1) B(K,1)];
 %break;
  %    end;
  %end;
end;
```

Programa que genera intercambios sin tomar en cuenta demandas de personal

```
function [a2,intercambio]=clas3(henf)
clc; format long;

[m n]=size(henf);
intercambio=[];
A=zeros(m,2);
B=zeros(m,2);



% Crea conjunto de índices de periodos en los que la enfermera está
 asignada para trabajarlos

for i=1:m
   k=1;
   if henf(i,1)==1
      A(i,k)=i;
      A(i,k+1)=1;
        k=k+2;
   else
```

```
      k=k+2;
   end;
end;

% Crea conjunto de índices de periodos en los que la enfermera no está
 asignada para trabajarlos

for i=1:m
   k=1;
   if (henf(i,1)==0)
      B(i,k)=i;
      B(i,k+1)=1;
        k=k+2;
   else
      k=k+2;
   end;
end;

[i1 j1]= find(A(:,1));
A=[i1 j1];
[i2 j2]= find(B(:,1));
B=[i2 j2];
[pa qa]=size(A);
[pb qb]=size(B);

if isempty(A)==0 && isempty(B)==0


   a2=ones(m,pa*pb);
    for i=1:m
        a2(i,:)=henf(i,1)*a2(i,:);
    end;

U=1;
sin=1;
%for sin=1:pa*pb
while sin<=(pb*pa)
for K=1:pb
        for K1=1:pa
            a2(B(K,1),sin)=1;
            a2(A(K1,1),sin)=0;
            intercambio(U,:)=[A(K1,1) B(K,1)];
          U=U+1;
          sin=sin+1;
    %        break;
```

```
      end;
     %break;
  end;
break;
end;
end;
```

Programa que determina el costo de los horarios

```
function [nc,c]=costosfin(ncol,ws,maxvio,pref,inter)
clc; format long;

[n m]=size(ncol');
ncol1=ncol';
violaciones=zeros(n,1);
r=zeros(n,1);
r2=zeros(n,1);
nc=[];
c=[];


[p1 q1]=size(inter);

%Se cuantifican violaciones individuales por patrones indeseables:
% 000 100 000, 000 010 000, 000 001 000

for i=1:n
    if inter(i,2)==1
    elseif inter(i,2)==2
    elseif inter(i,2)==3
    elseif inter(i,2)==m-2
    elseif inter(i,2)==m-1
    elseif inter(i,2)==m
    else
        s=mod(inter(i,2),3);
        if s==1
            if
 ncol1(i,inter(i,2)-1)==0&&ncol1(i,inter(i,2)-2)==0&&ncol1(i,inter(i,2)-
3)==0&&ncol1(i,inter(i,2)+3)==0&&ncol1(i,inter(i,2)+4)==0&&ncol1(i,inter(
i,2)+5)==0
                violaciones(i)=violaciones(i)+pref(1,1);
            end;
        elseif s==2
            if
```

```
 ncol1(i,inter(i,2)-2)==0&&ncol1(i,inter(i,2)-3)==0&&ncol1(i,inter(i,2)-
4)==0&&ncol1(i,inter(i,2)+2)==0&&ncol1(i,inter(i,2)+3)==0&&ncol1(i,inter(
i,2)+4)==0
                violaciones(i)=violaciones(i)+pref(1,2);
            end;
        elseif s==0
            if
 ncol1(i,inter(i,2)-3)==0&&ncol1(i,inter(i,2)-4)==0&&ncol1(i,inter(i,2)-
5)==0&&ncol1(i,inter(i,2)+1)==0&&ncol1(i,inter(i,2)+2)==0&&ncol1(i,inter(
i,2)+3)==0
                violaciones(i)=violaciones(i)+pref(1,3);
            end;
        end;
    end;
end;


%Determina si la enfermera trabaja o no el día j
for i=1:n
   j=1;
   m1=1;
   while j< m-1
          if  (ncol1(i,j)==1)||(ncol1(i,j+1)==1)||(ncol1(i,j+2)==1)
             g(i,m1)=1;
             m1=m1+1;
          else
            m1=m1+1;
          end;
          j=j+3;
      end;
   end;

%Se determina aversión por el número de días consecutivos

for i=1:n
        %for j=1:(m/3)-1
        j=1;
        while j<(m/3)-1
            if(g(i,j)==1)&&(g(i,j+1)==1)
            r(i,1)=r(i,1)+1;
          j=j+1;
            else
                if r(i,1)>=ws;%-1;
                r2(i,1)=1;
                j=j+1;
```

```
                end;
                    r(i,1)=0;
                    j=j+1;
        end;
    end;
    if r(i,1)>=ws;%-1;
        r2(i,1)=1;

    end;
end;


 for i=1:n
 if r2(i,1)==1
     violaciones(i)=violaciones(i)+pref(1,4);
 end;
end;

 %Se determina costo por la aversión al patrón DN
for i=1:n
    for j=1:m-3
        if mod(j,3)==1
             if ncol1(i,j)==1&&ncol1(i,j+5)==1
                 violaciones(i)=violaciones(i)+pref(1,5);
             end;
        else
        end;
    end;
end;

%Se determina violación por patrón ND
for i=1:n
    for j=1:m-3
        if mod(j,3)==0
             if ncol1(i,j)==1&&ncol1(i,j+1)==1
                 violaciones(i)=violaciones(i)+10;
             end;
        else
        end;
    end;
end;

%Se eliminan columnas que tengan un número de violaciones mayor al
%permitido
k=1;
```

```
for i=1:n
   if violaciones(i)<maxvio
      nc=[nc;ncol1(i,:)];
      c(k,1)=2^(violaciones(i));
      k=k+1;
   end;
end;
nc=nc';
```

Programa que genera las columnas alternativas para cada enfermera

```
function
[ncol,c3,htabenf]=gencol2(hbase,henf,c,ws,qMIN,qMAX,dayoff,dl,maxvio,maxc
ol,tc,tcenf,fs,d1,htabenf,pref)
clc; format long;

nuevacol=[];
ncol=[];
columprom=mean(tc);
c3=[];

if tcenf >= columprom
   [nuevacol inter]=clasprom3(hbase,henf,qMIN,qMAX);
  elseif tcenf < columprom
   [nuevacol inter]=clas3(henf);
end;
if isempty(nuevacol)==0
    %Se verifica que los horarios generados no violen restricciones
 fuertes
    nuevacol=revleg1(nuevacol,c,dayoff,dl,ws,inter);
    %Se calcula el costo de asignación de los horarios generados según
 las
    %violaciones en las que incurren.
    if isempty(nuevacol)==0
        [nuevacol c1]=costosfin(nuevacol,ws,maxvio,pref,inter);
        %Se selecciona el mínimo de columnas entre máxcol y las
 columnas generadas
       if isempty(nuevacol)==0
           [m n]=size(nuevacol);
           t=min(n,maxcol);
           p = randperm(n);
           for i=1:t
               k(1,i)=p(1,i);
           end;
```

```
            r=[];
            %%%%VER COSTOS
            c2=[];
            for j=1:t
                i=k(1,j);
                r=[r nuevacol(:,i)];
                c2=[c2;c1(i,1)];
            end;
            if isempty(r)==0
                [nuevacol htabenf c2]=redun2(r,htabenf,c2,fs,d1);
                if isempty(nuevacol)==0
                    ncol=nuevacol;
                    c3=c2;
                end;
            end;
        end;
    end;
end;
```

Programa que determina el número máximo de violaciones

```
function [maxvio]=maxviol(violaciones);
clc; format long;

maxvio=inf*ones(size(violaciones));
[i,j]=size(violaciones);
vmedia=mean(violaciones);
vdesv=std(violaciones);
for k=1:i%cambiepori
   if (violaciones(k,1))>(vmedia+vdesv)
      maxvio(k,1)=floor(max(1,vmedia-vdesv));
   end;
end;
```

Programa que resuelve los problemas enteros generados en cada iteración

```
function
 [z,fval,exitflag,bandera,z1]=opbin(A,rest,qMIN,qMAX,qrest,costo,bandera)
clc;

 [qi qj]=size(rest);
s=qMAX-qMIN;
[i2 j2]=size(s);
sdp=[];
```

```
%Transformación de variables enteras a variables binarias

sdp=eye(i2,i2);

ydp=zeros(i2,4*i2);

k=1;
for i=1:i2
    ydp(i,k)=1;
    ydp(i,k+1)=2;
    ydp(i,k+2)=4;
    ydp(i,k+3)=6;
    k=k+4;
end;

abin=[-sdp A ydp;zeros(qi,i2) rest zeros(qi,i2*4)];

b=[qMIN;qrest];

costoydp=ones(i2*4,1);

i=1;
while i<=(4*i2-3)
    costoydp(i,1)=1;
    costoydp(i+1,1)=2;
    costoydp(i+2,1)=4;
    costoydp(i+3,1)=6;
i=i+4;
end;

c=[zeros(i2,1);costo;50*costoydp];

[h fval exitflag output]=bintprog(c,[],[],abin,b);

%Se cuenta el numero de enfermeras externas
 i=(i2+qj+1);
 [p q]=size(h);

 while i<(p-4)
     for k=1:i2
       ydp1(k)=h(i)+h(i+1)+h(i+2)+h(i+3);
    i=i+4;
    end;
```

```
    end;

%Horarios seleccionados
z=[];
z1=[];
for k=(i2+1):(i2+qj)
 if h(k)==1
  z=[z A(:,k-i2)];
  z1=[z1 costo(k-i2,1)];
 end;
end;
```

Programa que verifica la optimalidad del horario

```
function [o,violaciones]=optimo2(a,h,ws,dayoff,c,qMIN,qMAX,pref)
clc; format long;

o=0;
[n,m]=size(h);
%d=zeros(n,1);
g=zeros(n,m/3);
r=zeros(n,1);
r2=zeros(n,1);
optim=zeros(n,4);
z=zeros(m,1);
v=zeros(m,2);
violaciones=zeros(n,1);

%Mínimo de días que debe laborar cada enfermera
for i=1:n
    min=nnz(h(i,:));
    if  min<c
        violaciones(i)=violaciones(i)+10;
    end;
end;

%Determina si la enfermera trabaja o no el día j
for i=1:n
    j=1;
    m1=1;
    while j< m-1
        if  (h(i,j)==1)||(h(i,j+1)==1)||(h(i,j+2)==1)
            g(i,m1)=1;
            m1=m1+1;
        else
```

```
            m1=m1+1;
        end;
        j=j+3;
    end;
end;
%Verifica que se cumplan los requerimientos de las enfermeras
   for i=1:n
       if dayoff(i,1)==0
       elseif g(i,(dayoff(i,1)))==1
           optim(i,3)=1;
           violaciones(i)=violaciones(i)+10;
       end;
   end;

%verifica si se viola el número de días consecutivos
for i=1:n
    for j=1:(m/3)-1
        if(g(i,j)==1)&&(g(i,j+1)==1)
            r(i,1)=r(i,1)+1;
        else
            if r(i,1)>=ws;%-1;
               r2(i,1)=1;
            end;
            r(i,1)=0;
        end;
    end;
    if r(i,1)>=ws-1;
        r2(i,1)=1;
        %break;
    end;
end;

 for i=1:n
     if r2(i,1)==1
         optim(i,4)=1;
         violaciones(i)=violaciones(i)+pref(i,4);
     end;
 end;

  %Se determina costo por la aversión a los patrones 000 100 000 000
 010 000
  %000 001 000
for i=1:n
    for j=4:m-3
        if h(i,j)==1
```

```
        if mod(j,3)==1
             if
 h(i,j-1)==0&&h(i,j-2)==0&&h(i,j-
3)==0&&h(i,j+3)==0&&h(i,j+4)==0&&h(i,j+5)==0
                 violaciones(i)=violaciones(i)+pref(i,1);
             end;
        elseif mod(j,3)==2
             if
 h(i,j-2)==0&&h(i,j-3)==0&&h(i,j-
4)==0&&h(i,j+2)==0&&h(i,j+3)==0&&h(i,j+4)==0
                 violaciones(i)=violaciones(i)+pref(i,2);
             end;
        elseif mod(j,3)==0
             if
 h(i,j-3)==0&&h(i,j-4)==0&&h(i,j-
5)==0&&h(i,j+1)==0&&h(i,j+2)==0&&h(i,j+3)==0
                 violaciones(i)=violaciones(i)+pref(i,3);
             end;
        end;
        end;
    end;
end;


 %Se determina costo por la aversión al patrón DN
for i=1:n
    for j=1:m-3
        if mod(j,3)==1
             if h(i,j)==1&&h(i,j+5)==1
                 violaciones(i)=violaciones(i)+pref(i,5);
             end;
        else
        end;
    end;
end;

%Se determina violación por patrón ND
for i=1:n
    for j=1:m-3
        if mod(j,3)==0
             if h(i,j)==1&&h(i,j+1)==1
                 violaciones(i)=violaciones(i)+10;
             end;
        else
        end;
```

```
    end;
end;

%verifica si se satisface la demanda por periodo
   k=1;
  for j=1:m/3
     [I,J,K]=find(a(:,j)=='D');
      z(k,1)=sum(K);
      [I,J,K]=find(a(:,j)=='T');
      z(k+1,1)=sum(K);
        [I,J,K]=find(a(:,j)=='N');
      z(k+2,1)=sum(K);
      k=k+3;
  end;

  for i=1:m
   if z(i,1)<qMIN(i,1)
      v(i,1)=1;
   elseif  z(i,1)>qMAX(i,1)
               v(i,2)=1;
   end;
  end;

  %'Horario no óptimo';

if (nnz(optim)~=0)||(nnz(v)~=0)||(nnz(violaciones)~=0)
    o=1;
end;
```

Programa que verifica si los horarios generados son redundantes o no

```
function [ncol2,htabenf,c1]=redun2(ncol,htabenf,c,fs,d1)
clc; format long;

ncol1=ncol';
[n m]=size(ncol1);
ncol2=[];
c1=[];

[colindex]=redunindex2(ncol1,fs,d1);

k=1;
for j=1:n
    [p q]=find(htabenf==colindex(j));
    if isempty(p)==1
```

```
        [p1 q1]=find(htabenf,1,'last');
        if isempty(q1)==1
          htabenf(1,1)=colindex(j);
        ncol2=[ncol2;ncol1(j,:)];
          c1(k,1)=c(j,1);
          k=k+1;
        elseif isempty(q1)==0
          htabenf(1,q1+1)=colindex(j);
          ncol2=[ncol2;ncol1(j,:)];
          c1(k,1)=c(j,1);
          k=k+1;
        end;
    end;
end;
ncol2=ncol2';
```

Programa que genera el indicador para cada horario

```
function [colindex]=redunindex2(ncol1,fs,d1)
clc; format long;
[n m]=size(ncol1);
g=zeros(n,m/3);
colind=[];
colindex=[];

fss=[];

%Determina qué turno trabaja la enfermera el día j
for i=1:n
   j=1;
   m1=1;
   while j< m-1
          if  (ncol1(i,j)==1)
             g(i,m1)=1;
             m1=m1+1;
          elseif ncol1(i,j+1)==1
             g(i,m1)=2;
             m1=m1+1;
          elseif ncol1(i,j+2)==1
             g(i,m1)=3;
             m1=m1+1;
          elseif (ncol1(i,j)==0)&&(ncol1(i,j+1)==0)&&(ncol1(i,j+2)==0)
             g(i,m1)=0;
             m1=m1+1;
          end;
          j=j+3;
```

```
   end;
end;


for i=1:n
    for j=1:(m/3)
        if g(i,j)==1
            fss(i,j)=fs(1);
        elseif g(i,j)==2
            fss(i,j)=fs(2);
        elseif g(i,j)==3
            fss(i,j)=fs(3);
        elseif g(i,j)==0
            fss(i,j)=fs(4);
        end;
    end;
end;
for i=1:n
    for j=1:(m/3)-1
        colind(i,j)=(fss(i,j)+d1(j))*(fss(i,j+1)+d1(j+1));
    end;
end;


for i=1:n
   colindex(i)=sum(colind(i,:));
end;
```

Programa que determina si los horarios generados son factibles o no

```
function [A]=revleg1(ncol,c,dayoff,dl,ws,inter)
clc; format long;

[q1 p1]=size(ncol);

g=zeros(p1,q1/3);
optim=zeros(p1,5);

r=zeros(p1,1);
r2=zeros(p1,1);
r3=zeros(p1,1);
r4=zeros(p1,1);

%Mínimo de días que debe laborar cada enfermera
for i=1:p1
   min=nnz(ncol(:,i));
```

```
          if  min<c
         optim(i,1)=1;
      end;
end;

%Determina si la enfermera trabaja o no el día j
ncol1=ncol';
for i=1:p1
   j=1;
   m1=1;
   while j< q1-1
          if  (ncol1(i,j)==1)||(ncol1(i,j+1)==1)||(ncol1(i,j+2)==1)
             g(i,m1)=1;
             m1=m1+1;
          else
            m1=m1+1;
          end;
          j=j+3;
      end;
   end;

   %Verifica que se cumplan los requerimientos de las enfermeras

   for i=1:p1
      if dayoff~=0
       if g(i,dayoff)==1
         optim(i,2)=1;
         end;
      end;
      end;

     %Verifica que se cumplan el minimo numero de dias libres
     for i=1:p1
     if sum(g(i,:))== q1/3
         optim(i,3)=1;
     end;
     end;

     %Verifica que no se viole el maximo numero de dias consecutivos a
     %laborar

 for i=1:p1
    for j=1:(q1/3)-1
        if(g(i,j)==1)&&(g(i,j+1)==1)
            r(i,1)=r(i,1)+1;
```

```
        else
            r(i,1)=0;
        end;
    end;
    if r(i,1)>=ws;
        r2(i,1)=1;

    end;
end;

for i=1:p1
     if r2(i,1)==1
         optim(i,4)=1;
     end;
 end;


 %Se verifica que los horarios generados no tengan patrones indeseables

 for i=1:p1
    if mod(inter(i,2),3)==1
        if inter(i,2)==1
            if ncol1(i,inter(i,2)+1)==1||ncol1(i,inter(i,2)+2)==1
                optim(i,5)=1;
            end;
        else
            if
 ncol1(i,inter(i,2)-
1)==1||ncol1(i,inter(i,2)+1)==1||ncol1(i,inter(i,2)+2)==1
                optim(i,5)=1;
            end;
        end;

    end;

    if mod(inter(i,2),3)==2
        if ncol1(i,inter(i,2)-1)==1||ncol1(i,inter(i,2)+1)==1
            optim(i,5)=1;
        end;

    end;

    if mod(inter(i,2),3)==0
        if inter(i)==q1
            if ncol1(i,inter(i,2)-1)==1||ncol1(i,inter(i,2)-2)==1
```

```
                optim(i,5)=1;
            end;
        else
            if
 ncol1(i,inter(i,2)-1)==1||ncol1(i,inter(i,2)-
2)==1%||ncol1(i,inter(i,2)+1)==1
                optim(i,5)=1;
            end;
        end;

    end;
end;

     %Elimina las columnas que no satisfacen las restricciones fuertes

 A=[];

 for i=1:p1
   if nnz(optim(i,:))==0
     HENF=['A=[A ncol(:,',int2str(i),')];'];
     eval(HENF)

end;
end;
```

Glosario

El objetivo de éste glosario es definir conceptos particulares a la organización de horarios de enfermeras y mencionar conceptos que parecen relevantes de considerar para clarificar los tema expuestos. El orden de las definiciones es el considerado para facilitar la comprensión de éstas.

Turno

Periodo de tiempo predefinido de antemano, el cual tiene un inicio y un término que son fijos. En dicho periodo los miembros del personal son o no asignados a tareas. Cada hospital define sus tipos de turnos de acuerdo a sus necesidades específicas. Generalmente en los hospitales se establecen turnos de ocho o de doce horas, lo que da lugar a tipos de turnos como los que se muestran en la figura 1.

	Tipo de turno	Inicio	Término
	8 horas		
M	Turno de la mañana	06:45	14:45
T	Turno de la tarde	14:30	22:00
N	Turno de la noche	22:00	07:00
	12 horas		
AM	Turno AM	08:00	20:00
PM	Turno PM	20:00	08:00

Fig. 1 Tipos de turnos

Periodos de demanda

Este concepto se definió inicialmente para extender el modelo básico que contempla sólo turnos de 8 horas a un modelo que contemplara turnos de 8 y de 12 horas simultáneamente en la organización de horarios.
Estos se definen como bloques de tiempo que se usan para contrarrestar la posibilidad de traslapar horas de trabajo de dos turnos diferentes y disminuir además el número de renglones del modelo. Es decir, el día se divide en un conjunto de periodos contiguos de diferente longitud. Cada turno de los especificados en el problema puede contener uno o más de estos periodos.

Si se toman en cuenta turnos de ocho y de doce horas se tienen en total cinco turnos diferentes: tres de ocho horas que son el turno de la mañana, el turno de la tarde y el turno de la noche y dos de doce horas que se denominan como turno AM y turno PM.

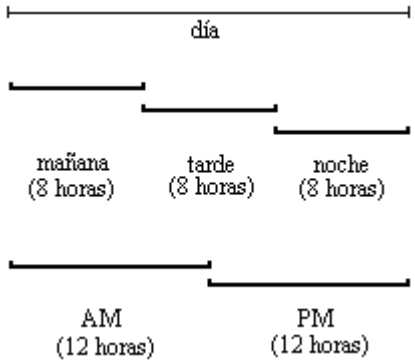


Fig. 2 Tipos de turnos

Así, los periodos de demanda para un modelo que contempla turnos de ocho y de doce horas están dados por dos periodos de ocho horas y dos de cuatro como se muestra en la figura 3.

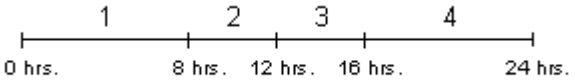


Fig. 3 Periodos de demanda para turnos de 8 y de 12 horas

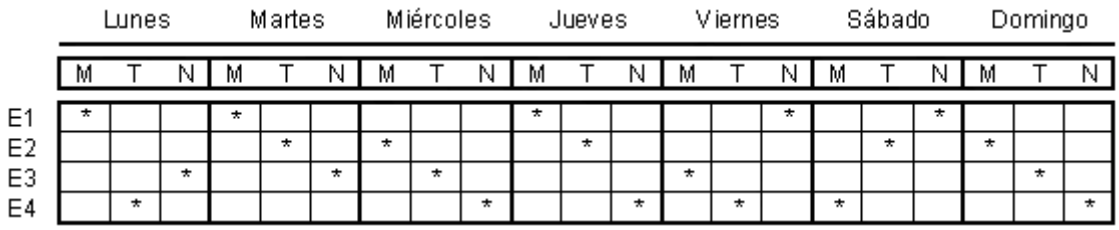
Asignación

Una asignación corresponde a determinar qué horario laborará cada enfermera, para ello, previamente en cada horario se ha designado cuales turnos se trabajaran y cuales turnos no durante el periodo de planeación que se tenga contemplado.

Como ejemplo se considerará un conjunto de cuatro enfermeras, el horizonte de planeación es de una semana y se toman en cuenta tres turnos (mañana, tarde y noche). La figura 4 muestra un ejemplo de una asignación a cada una de las cuatro enfermeras.

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
E1	M	M	-	M	N	N	-
E2	-	T	M	T	-	T	M
E3	N	N	T	-	M	-	T
E4	T	-	N	N	T	M	N

Fig. 4



(*) Denota que se laborará el turno indicado

Fig. 5 Asignación de un horario a cuatro enfermeras considerando tres turnos

Horario

Un horario corresponde a los turnos que deberá cubrir cada enfermera diariamente sobre el horizonte de planeación que se establece. En el modelo, los horarios están representados por columnas cuyas componentes están dadas por ceros o unos. Es decir, si el horario contempla laborar determinado turno $t \in T$ de cierto día $d \in D$, la componente correspondiente será igual a 1, de no ser así la componente será igual a cero. Así un horario *h* está dado por la columna:

$$h = (h_{11}, h_{12}, h_{13}, h_{14}, \ldots, h_{21}, h_{22}, \ldots, h_{dt})^T .$$

En donde h_{dt} es igual a cero o uno, para toda $d \in D$ y $t \in T$.

Por ejemplo, la columna que se muestra a continuación representa un horario en donde se establecen los primeros cuatro días de trabajo de una cierta enfermera, en donde se establece que laborará el primer turno de esos días.

$$(100\ 100\ 100\ 100)^T$$

Calidad de un horario

Percepción individual de las enfermeras sobre qué tanto el horario que les es asignado se ajusta a sus deseos de no trabajar cierto día, de laborar cierto periodo determinado día o sobre otras propiedades del horario como laborar determinados patrones de trabajo o el número de días consecutivos a laborar.

Problema de cobertura "Set Covering Problem"

El problema de cobertura es un problema de programación entera cuyos valores de coeficientes y variables los conforman vectores cuyos elementos son ceros y unos.

Sea *J* un subconjunto de columnas y x_j una variable binaria que toma el valor de 1, si *j* pertenece a *J*, y cero en otro caso.

La estructura formal del problema es la siguiente:

$$\begin{aligned} &Min\ \ cx \\ &s.a.\quad Ax \geq b \\ &\qquad x_j = 0 \text{ ò } 1 \quad j = 1,...,n \end{aligned}$$

.

En donde, *A* es una matriz de (*m* x *n*) cuyas entradas son cero o uno, *c* vector de (1 x *n*) es el renglón de coeficientes de costo cuyas componentes son positivas, x de dimensión (*n* x 1) es un vector de variables cero o uno, y *b* es un vector de unos de tamaño (*m* x 1).
El problema consiste en "cubrir" las filas de la matriz A con un subconjunto de columnas a mínimo costo. En el problema se tiene que la columna *j* puede cubrir la fila *i*, si a_{ij} = 1. El objetivo es obtener un conjunto de columnas, de mínimo costo, en el que al menos una columna cubra una fila. La restricción $Ax \geq b$ garantiza que cada fila de la matriz sea cubierta por al menos una columna en la solución.

Debido a la particular estructura de este problema, este tipo de modelo se ha utilizado para plantear muchos problemas reales, entre los cuales, se encuentra el problema de la organización de horarios.

El modelo que se desea presentar plantea al problema de organización de horarios como un problema de cobertura en donde las columnas representan horarios alternativos que una enfermera puede tomar sobre el horizonte de planeación y los renglones los turnos diarios que se trabajaran o no para satisfacer la demanda de personal del hospital.

Restricciones fuertes

Una restricción fuerte es aquella que implica factibilidad de la solución, es decir, son restricciones que deben de ser cumplidas estrictamente y no pueden ser violadas bajo ninguna circunstancia. En el caso de la organización de horarios para enfermeras algunos ejemplos son: se satisfaga la demanda de personal o que el número de días que debe laborar cada enfermera sea el asignado.

Ejemplos:

1. Todos los turnos de todos los días en el horizonte de planeación deben de ser asignados a un miembro del personal de enfermería.
2. Una enfermera no puede ser asignada a dos turnos en un mismo día.
3. Debe de existir un descanso de 16 horas entre turnos.
4. El número de días a trabajar en el horizonte de planeación no puede ser menor al especificado al inicio de la planeación.

Restricciones suaves

Una restricción suave es aquella que en términos ideales se desearía satisfacer, pero no siempre es posible en términos reales. A cada restricción suave se le asigna un costo por

ser violada. Mediante la asignación de éste costo se determina cuáles restricciones se quieren satisfacer más y a cuales no se les da tanta importancia.
Un punto importante de estas restricciones es que son utilizadas para determinar la "calidad" de los horarios factibles. En la práctica es realmente excepcional hallar un horario que satisfaga todas las restricciones suaves por lo que el objetivo primordial es el de minimizar las penalizaciones por las violaciones a estas restricciones.

Ejemplos:

1. Número de días consecutivos a laborar.
2. Número de días consecutivos libres.
3. Número consecutivo de turnos del mismo tipo.
4. Máximo número de asignaciones de cada turno por semana.
5. Número de días a trabajar en fines de semana.
6. Asignaciones a laborar fines de semana completos.

En el caso de las restricciones 1 a 5 se considerarán penalizaciones cuando se exceda de los valores establecidos.

Relajaciones

Una relajación es un subproblema que surge a partir de ignorar ciertas restricciones de un problema. El objetivo es obtener otro cuya solución es por lo menos más fácil de hallar que en el original. Una utilidad de estas relajaciones radica en que proporcionan cotas a la solución del problema original.

Existen muchas formas de relajar un problema, por ejemplo una relajación en un problema de programación entera consiste en ignorar la restricción concerniente a que en la solución las variables sean enteras, permitiendo que éstas puedan tomar cualquier valor.

De manera más formal se tiene[4]:

Una relajación del problema entero

$$\begin{aligned} &Max\ cx \\ &\text{s.a. } x \in S \end{aligned}$$

donde S es el conjunto de soluciones factibles.

Es cualquier problema de maximización

$$\begin{aligned} &Max\ Z_{PR}(x) \\ &\text{s.a. } x \in S_{PR} \end{aligned}$$

con las siguientes dos propiedades :

[4] Nemhauser, Rinnooy, y Todd (editores), "Handbooks in Operations Research and Management Science".

i. $S \subseteq S_{PR}$
ii. $cx \leq Z_{PR}(x)$ para $x \in S$

Optimización multiobjetivo

El problema de optimización multi-objetivo también llamado de optimización multicriterio o problema de optimización vectorial es un problema que consiste en encontrar un vector de variables de decisión el cual satisfaga las restricciones del problema y que optimice un vector función cuyos elementos representan funciones objetivo. El problema radica en que los criterios antes mencionados se encuentran en conflicto unos con otros y todos deben de satisfacerse simultáneamente e incluso optimizarse.

Planos cortantes *"cutting planes"*
Las técnicas basadas en planos cortantes consisten en encontrar la solución de un problema de programación entera a partir de la solución de una secuencia de relajaciones del problema original. La primera relajación y la más sencilla, consiste en quitar la restricción entera del problema. Al resolver la relajación del problema como un problema de programación lineal se tienen dos opciones. Si la solución óptima es entera también es óptima para el problema original y la solución deseada se ha encontrado. De no ser así, a partir de ella se obtiene una nueva restricción, también llamada corte, que se agregará al conjunto de restricciones de la primera relajación, con lo que se obtiene una nueva relajación. En cada iteración, es decir, cada vez que se encuentra la solución a la relajación en turno se va mejorando el resultado hasta obtener la solución óptima al problema entero si es que ésta existe. El corte cumple con dos propiedades: la primera es que cualquier solución factible entera satisface el corte y la segunda es que la solución fraccional de la cual se obtuvo el corte no lo satisface.

Satisfacción de restricciones "Constraint Satisfaction"

Una restricción es una relación lógica entre varias variables, las cuales toman valores de un dominio dado. Una restricción representa información parcial sobre la variable de interés, de hecho restringe los posibles valores que ésta variable puede tomar. Estas restricciones aparecen de manera natural como formas de expresar problemas de cualquier campo (Barták). Por ejemplo, la suma de los tres ángulos interiores de un triángulo tienen que sumar 180 grados, la hora de llegada a un determinado lugar no puede pasar de las tres de la tarde, el número de maletas de los pasajeros de un autobús no puede exceder de 80, etc. Las restricciones pueden ser condiciones, propiedades o requerimientos que se les piden a las variables en cuestión.

El problema de la satisfacción de restricciones es un problema en donde se cuenta con un conjunto de variables X_1, X_2 ,…, X_n y un conjunto de restricciones R_1, R_2,…, R_m. Cada restricción del problema restringe la combinación de valores que un conjunto de variables puede tomar. Cada variable X_i tiene un dominio D_i finito de posibles valores. El problema

consiste en encontrar valores de las variables X_i que satisfagan todas las restricciones simultáneamente, es decir, se trata de encontrar lo que se conoce como una asignación.

Cuando se trabaja con problemas de satisfacción de restricciones existen tres objetivos que se pueden perseguir: uno es encontrar una solución cualquiera que satisfaga las restricciones del problema, otra opción es encontrar todas las soluciones del problema y por último los problemas que involucran la optimización de una función objetivo o por lo menos una aproximación al óptimo, en cuyo caso se le denomina como problema de optimización de restricciones “Constraint Optimisation Problem”.

Entre los problemas que se resuelven con la ayuda de la satisfacción de restricciones se encuentran la asignación de recursos, la administración de fuerza de trabajo, la organización de cursos escolares y la organización de horarios para enfermeras.
Existen tres formas principales de resolver el problema de la satisfacción de restricciones: las técnicas basadas en algoritmos de búsqueda sistemática, como el método de generar y probar y el método de “backtracking”, las basadas en propagación de restricciones y los métodos que emplean la combinación de estas dos técnicas o variantes de éstas.

Bibliografía

Artículos:

Ahuja, R.K., O. Ergun, J.B. Orlin, A.P. Punnen. (2002), A survey of very large-scale neighborhood search techniques. Discrete Applied Mathematics 123, pp. 75-102.

Bard, Jonathan F. y Haidi W. Purnomo (2005), " Preference Scheduling for nurses using column generation", European Journal of Operational Research. EJOR, (2005), 164: 510-534.

Bruker, Peter, Edmund Burke, Rong Qu y Gerhard Post , "A Decomposition, Construction and Post-processing Approach for Nurse Rostering", en *Proceedings Multidisciplinary International Conference on Scheduling : Theory and Applications*, pp. 397-406, New York.

Benli, Ömer S. (1999), "The Branch and Bound Aproach", International Journal of Industrial Engineering, recuperado en: http://www.ie.bilkent.edu.tr/~omer/research/bb.html

Burke E., De Causmaecker P., Petrovic S. and Vanden Berghe G.: Variable Neighborhood Search for Nurse Rostering Problems, en: Metaheuristics: Computer Decision-Making (Combinatorial Optimization Book Series), Kluwer, (2003), pp. 153-172.

Coello, Coello Carlos A. (2002),"Metaheuristics for Multiobjective Optimization", CINVESTAV/IPN, México.

Coello, Carlos A. (1995), Introducción a los Algoritmos Genéticos, *Soluciones Avanzadas. Tecnologías de Información y Estrategias de Negocios*, Año 3, Número 17, pp. 5-11.

Cheang B., Li H., Lim A. and Rodrigues B. (2003), Nurse Rostering Problems - a Bibliographic Survey. EJOR, (2003) 151: 447-460.

Desrosiers, Jacques et. al. (2005), "A Primer in Column Generation", a publicar en Column Generation , M.M. Solomon eds. Springer.

Desrosiers, Jaques y Marco E. Lübbecke(2005), " Selected Topics in Column Generation", Operations Research,Vol. 53, No. 6, pp. 1007-1023.

Dowsland, Kathryn A. y Belarmino Adenso Díaz (2001), "Diseño de Heurísticas y Fundamentos del Recocido Simulado", Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No.20, pp. 34-52.

Dowsland K.: Nurse Scheduling with Tabu Search and Strategic Oscillation. EJOR, (1998) 106: 393-407.

Garfinkel S. Robert (1971), An Improved Algorithm for the Bottleneck Assignment Problem, The University of Rochester, N.Y., *Operations Research*, Vol. 19, No. 7, pp. 1747-1751.

Glover, Fred y Manuel Laguna (1997), *Tabu Search*, Kluwer Academic Publishers.

Gunawan, Aldy , Kien Ming, Kim Leng (2006), A Mathematical Programming Model for a Timetabling Problem, The 2006 world congress in computer science, computer engineering and applied computing. Nevada, USA.

Maisels , Amnon y Andrea Schaerf (2003) , "Modeling and Solving Employee Timetabling", Annals of Mathematics and Artificial Intelligence, Vol. 39, No. 1-2, pp. 41-59.

Nemhauser, George, et. a.l. (1998) " Branch and Price : Column Generation for Solving Huge Integer Programs. *Operations Research*, Vol. 46, No. 3, pp. 316-329.

Nonobe K. and Ibaraki T.: A Tabu Search Approach to the Constraint Satisfaction Problem as a General Problem Solver. EJOR, (1998) 106: pp. 599-623.

Pardalos Panos M. y Henry Wolkowicz (1994), Quadratic Assignment and Related Problems, DIMACS Series , American Mathematical Society, Vol. 16., pp. 1-42.

Ramalhinho Dias (2004), Métodos de solución de problemas de asignación de recursos sanitarios, Documentos de Trabajo, Fundación BBVA., Universidad Pompeu Fabra.

Randhawa, S.U., Sitompul, D., 1993. A heuristic-based computerized nurse scheduling system. Computer & Operations Research 20 (8), 837–844.

Rodríguez, Ericka Z. (2006), Asignación multicriterio de tareas a trabajadores, Tesis doctoral, Instituto de Organización y Control de Sistemas Industriales, Universidad Politécnica de Catalúña.

Schaerf, Andrea (1996), Local Search Techniques for Large High-School Timetabling Problems, IEEE Transactions on Systems, Man and Cybernetics.

Vanhoucke, Mario y Broos Maenhout (2005), Characterization and Generation of Nurse Scheduling Problem Instances, Ghent University, Ghent, Bélgica.

Warner, D. Michael y Juan Prawda (1972), A Mathematical Programming Model For Scheduling Nursing Personnel in a Hospital, Management Science, Vol. 19, No. 4. pp 411-422.

Libros:

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, New Jersey.

Bazaraa, M.S. et al. (1990) Linear Programming and network flows, segunda edición, John Wiley and Sons.

Chvátal Vasek (1983), "Linear Programming", Freeman, USA. 478 p

Gass, Saull(1985), Linear Programming, Methods and Applications, Mc Graw-Hill, 5ta. edición, pp. 343-349.

Greenberg, Harold (1971), Integer Programming, Academia Press, USA.

Hadley, G. (1962), "Linear Programming", Adisson-Wesley, USA., pp. 400-414.

Hincapié Isaza, et. al (2004), Técnicas heurísticas aplicadas al problema del cartero viajante, Scientia et Técnica, Grupo de Investigación en Planeamiento de Sistemas Eléctricos. Universidad Tecnológica de Pereira. Colombia No. 24.

Jauffred, M. (1971), "Métodos de optimización: programación lineal y gráficas", México. 720 p.

Murtagh, Bruce, "Advanced Linear Programming", Mc GrawHill, pp.90-93

Orchard, Hays (1968), "Advanced Linear Programming Computing techniques", Mc Grawhill, USA, pp. 239-270.

Rayward-Smith, V.J. ed. (1996), "Modern Heuristic Search Methods", John Wiley and Sons, pp. 1-25.

Sierksma, Gerard, "Linear and Integer Programming. Theory and Practice", Dekker, USA. 633 p.

Wu Nes (1981), "Linear Programming and Extensions", Mc GrawHill, pp. 235-255.

Wolsey A, Lourence (1998), "Integer Programming", Wiley-Interscience, USA.