UNIVERSIDAD AUTONOMA DE NUEVO LEON

FACULTAD DE INGENIERIA MECANICA Y ELECTRICA

DIVISION DE ESTUDIOS DE POSGRADO

UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARAMETROS DE ENTRADA

TESIS

EN OPCION AL GRADO DE MAESTRO EN CIENCIAS
EN INGENIERIA DE SISTEMAS

PRESENTA:

JOSE FLORENTINO AUGUSTO MEDINA JACOBO

SAN NICOLAS DE LOS GARZA, N. L. MAYO DE 2005

UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARAMETROS DE ENTRADA

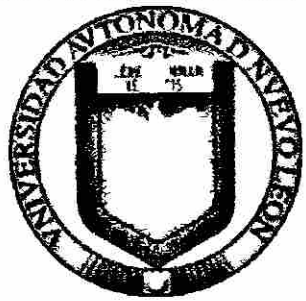
J.F.A.M.J.

2005

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO



UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARÁMETROS DE ENTRADA

TESIS

EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS EN

INGENIERÍA DE SISTEMAS

PRESENTA

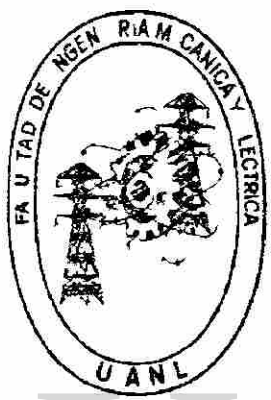
JOSÉ FLORENTINO AUGUSTO MEDINA JACOBO

SAN NICOLÁS DE LOS GARZA, N. L. MAYO DE 2005

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO



UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARÁMETROS DE ENTRADA

POR:

ING. JOSÉ FLORENTINO AUGUSTO MEDINA JACOBO

TESIS

EN OPCIÓN AL GRADO DE MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, N. L. **MAYO DE 2005**

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "**UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARÁMETROS DE ENTRADA**", realizada por el alumno José Florentino Augusto Medina Jacobo, matrícula 839990, sea aceptada para su defensa como opción al grado de Maestro en Ciencias en Ingeniería de Sistemas.

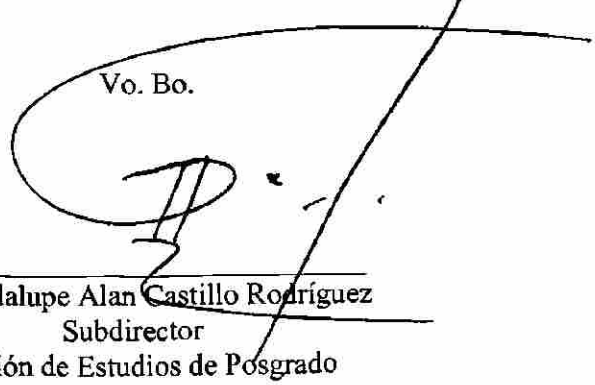
El Comité de Tesis

Dr. Ada M. Álvarez Socarrás
Asesor

Dr. Roger Z. Ríos Mercado
Revisor

Dr. César E. Villarreal Rodríguez
Revisor

Vo. Bo.



Dr. Guadalupe Alan Castillo Rodríguez
Subdirector
División de Estudios de Posgrado

DEDICATORIA

Quiero dedicar este trabajo de tesis con mi más sincero amor y respeto

a mi *Madre*:

Nemecia Jacobo Ramírez

Que con su sencillez, paciencia y sacrificio ha sabido brindarme la medicina para crecer como persona de bien en la vida.

a mi *Padre*:

José de Jesús Medina Ortiz

Por ser el sostén de esta familia y mantener la fe en mí.

a mis *Familiares*:

René Israel, Friede Viviana, Noham (El oso)**, Ruperta y Ma. Celsa**

Mi muy querida segunda familia.

a mi *Novia*:

Erika Ludivina Soto Castro

Que vino con su alegría a reactivar mis fuerzas. Gracias por tu paciencia y apoyo.

AGRADECIMIENTOS

Primero que todo quiero agradecer sinceramente a Dios por mantenerme en pie de lucha ante nuevos retos en mi vida.

Quiero agradecer infinitamente a mi asesora de tesis, la Dra. Ada M. Álvarez Socarrás, por confiar en mí y ser guía en este trabajo de tesis; por ser una mujer ejemplo y emprendedora de grandes retos en la vida.

A mis revisores, Dr. César Villarreal Rodríguez y Dr. Roger Z. Ríos Mercado, por sus comentarios y aportaciones a este trabajo de tesis.

A todo el personal docente del PISIS, al Dr. Oscar L. Chacón Mondragón y Dr. Igor S. Litvinchev, por ser mis maestros en este programa y ser parte fundamental de este desarrollo como profesionista.

Al proyecto CONACyT 36669-A por darme la oportunidad de participar como Asistente de Investigación y apoyarme económicamente para el estudio de la Maestría.

Quiero extender un agradecimiento muy especial a mis compañeros de estudio Mc. Conrado Borraz, Lic. Daniel Aguirre, Mc. Humberto J. Flores Villarreal, Ing. Angélica Salazar, Ing. Fernando Reyes, Lic. Gilberto Fuentes y Dr. Karim de Alba por brindarme su ayuda durante mi estancia en el PISIS.

A todos infinitamente gracias.

RESUMEN

José Florentino Augusto Medina Jacobo

Candidato para el Grado de Maestro en Ciencias en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Título del Estudio:

UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARÁMETROS DE ENTRADA

Número de páginas: 118

Objetivos y Método de Estudio

El desarrollo del presente trabajo se enfoca en un problema de diseño de red multiproducto con incertidumbre en parámetros de entrada. Se considera incertidumbre en dos de los parámetros del problema: costos de ruteo y demandas. La incertidumbre se modela vía escenarios discretos.

El problema de Diseño Robusto de Red Capacitada Multiproducto se modela a través de un programa entero mixto no lineal (MINLP, Mixed Integer Non-linear Program, por sus siglas en inglés). Desde el punto de vista de optimización, aún en los casos más sencillos, este problema es difícil de resolver ya que se ha demostrado que pertenece a la clase *NP-completo* [19, 33, 35, 56].

Reconociendo la dificultad para resolver este tipo de problemas, los objetivos de esta tesis son:

- Hacer un estudio de la estructura matemática del problema de diseño de red robusto, así como de problemas semejantes a este.
- Realizar un estudio amplio de las diferentes técnicas para resolver problemas bajo incertidumbre y una extensa investigación de trabajos referentes a características aleatorias en problemas.
- Desarrollar e implementar una metodología de búsqueda para el diseño robusto de red capacitada multiproducto.
- Generar una base de instancias para este tipo de problemas que sirvan de peldaño para investigaciones futuras por parte de otros investigadores, además que los resultados de esta metodología sean utilizados como comparativo contra trabajo futuros.
- Efectuar un estudio computacional exhaustivo de la metodología de búsqueda propuesta.

Con lo anterior se propone una metodología de búsqueda basada en estrategias de Descomposición de Benders [4] y Búsqueda Tabú [26] para tratar de encontrar soluciones robustas de buena calidad al problema de diseño robusto de red capacitada multiproducto.

Contribuciones y Conclusiones

Como resultado de este trabajo de investigación se entrega una metodología de solución estructurada en programas en lenguaje C y archivos denominados *batch's*. Como resultado del uso de esta metodología se entrega una solución robusta al problema de diseño robusto de red capacitada multiproducto, que consiste en un diseño de red que se encuentra a un *P%* de la cota superior del óptimo de cada escenario.

El análisis del problema de diseño de redes bajo incertidumbre permitió conocer las características propias de este problema así como las dificultades que se presentan al tratar de resolverlo. La representación computacional de una red brinda la oportunidad de aplicar este conocimiento a diferentes tipos de problemas relacionados, en particular en el área de telecomunicaciones.

Profesor asesor: 

Dr. Ada M. Álvarez Socarrás

ÍNDICE

CAPÍTULO 1
INTRODUCCIÓN

1.1 Descripción y Relevancia del Problema

La mayoría de las aplicaciones que se pueden encontrar en el amplio mundo de optimización e investigación de operaciones suponen que los datos para un problema dado son conocidos con exactitud. Asimismo, para muchos problemas actuales, los datos del problema no pueden ser conocidos con exactitud por una amplia variedad de razones. La primera razón es debido a una simple medida de error en los datos de entrada. La segunda y más fundamental es que algunos datos representan información acerca del futuro (ejemplo: la demanda de un producto o el precio para un período en el futuro) y simplemente no puede ser conocida con certeza.

Ahora bien, ante un mundo cambiante y más en problemas que se presentan en un ambiente industrial, económico, gubernamental, etc., los datos o parámetros de entrada a un problema en ocasiones presentan incertidumbre, esto es, que dichos datos pueden tomar diferentes valores sin saber con exactitud cuál de estos valores ocurrirá sobre un horizonte de tiempo determinado a futuro. Sobre esta temática es el desarrollo del presente trabajo.

Antes de entrar a detalle con la problemática que nos atañe, cabe hacer mención sobre la versión determinista del problema en estudio, el cual es el diseño de red óptima. Dicho problema se caracteriza por la búsqueda de la mejor configuración de red que satisfaga un conjunto dado de requerimientos. Aquí se pueden considerar problemas de expansión de red, problemas de ubicación de plantas, problemas de selección de procesos, etc.

En varias de las problemáticas mencionadas arriba, el diseñar consiste en seleccionar un subconjunto de nodos de la red en los cuales se ubicarán plantas y en asignar

clientes a esas plantas con el objetivo de minimizar los costos totales. Éste es el caso del llamado problema de ubicación de planta [11].

En otras aplicaciones, el problema consiste en la selección de un subconjunto de arcos de forma que se garantice cierto desempeño de la red y además se minimicen todos los costos totales por concepto de construcción (o utilización de arcos) y por transportación de bienes por los mismos.

En telecomunicaciones, por ejemplo en el diseño de red de supervivencia, se desea encontrar una red de mínimo costo que satisfaga ciertos requerimientos de conectividad, esto es, que entre cada par de nodos exista cierto número de caminos disjuntos [22].

En otro trabajo tenemos el problema de instalar o cargar, a un cierto costo, una de p tipos de instalaciones capacitadas. Aquí el objetivo es determinar la configuración de las instalaciones a cargar en cada arco de forma que se satisfaga la demanda de cierto producto al menor costo posible [51].

En cada uno de los problemas que mencionamos, la decisión central es "invertir o no invertir", "construir o no construir", la cual puede ser modelada imponiendo "cargas fijas" en arcos o nodos apropiados de la red.

El problema de diseño que será objeto de estudio en esta tesis consiste en lo siguiente:

Se tiene un conjunto de nodos, entre los cuales se distinguen varias parejas que constituirán respectivamente el origen y destino de cada uno de los productos que deben ser transportados por la red que hay que diseñar. Para ello se dispone de un conjunto de aristas potenciales, cada una con un costo fijo por su utilización y una capacidad determinada. También hay costos de transportación que dependen de la cantidad de producto que esté circulando por cada arista y en general es diferente para los distintos productos.

El objetivo es seleccionar las aristas que formarán parte de la red, de forma que se garantice la operación adecuada de la misma (transportación adecuada de los productos sin violar las capacidades de las aristas) a un costo total mínimo, el cual incluye costo por construcción y por operación.

En la versión determinista de este problema se considera que todos los parámetros de entrada (demandas de los productos, capacidades de las aristas, costos fijos y costos de transportación) se conocen con certeza.

En la literatura aparecen numerosos trabajos sobre la versión no capacitada del diseño de una red multiproducto, el indicar que el problema es no capacitado se refiere a que las aristas no poseen cotas de capacidad sobre lo que se puede transportar a lo largo de ellas. Dentro de estos trabajos podemos mencionar a Hoang [32]; Dionne y Florian [17]; Barr, Glover, y Klingman [3]; Magnanti y Wong [49]; Balakrishnan y Magnanti [2]; Holmberg y Migdalas [34].

En cuanto a redes con capacidades finitas en las conexiones, si se consideran redes orientadas, es decir, redes donde cada conexión potencial entre nodos tiene una sola orientación específica, el problema anterior ha sido estudiado por Khang y Fijiwara [41], Holmberg y Yuan [36, 37], y Crainic, Gendreau y Farvolden [9], por citar algunos. Sin embargo, si se consideran redes no orientadas, es decir, redes donde en cada arco puede transportarse producto en ambas direcciones, el problema ha sido menos tratado en la literatura. Podemos citar por ejemplo, Herrmann [30], y más recientemente trabajos de De-Alba, Álvarez y González-Velarde [12, 13, 15], De-Alba [14], Cobos-Zaleta y Álvarez [7], y Cobos-Zaleta [6].

Ahora bien, un menor esfuerzo ha sido dirigido aún al problema considerando incertidumbre en algunos parámetros, lo cual lo hace más realista pero a la vez más difícil de resolver y trae consigo considerables retos algorítmicos.

Sin duda uno de los aspectos fundamentales que introduce este trabajo de tesis es considerar incertidumbre en algunos de los parámetros al problema de diseño de red capacitada multiproducto descrito anteriormente. La incertidumbre la mostramos en dos de sus principales parámetros de entrada, como son las demandas de cada producto y los costos variables o de ruteo. Al problema bajo incertidumbre lo referiremos como Diseño Robusto de Red Capacitada Multiproducto.

La estrategia para modelar la incertidumbre que consideramos en este problema de diseño se basa en el uso de escenarios. Un escenario de datos de entrada específico

representa una realización potencial de parámetros importantes del modelo de decisión es decir, ocurre con alguna probabilidad positiva, no necesariamente conocida.

Existe una amplia variedad de criterios para diferenciar entre decisiones al considerar incertidumbre. Es posible convertir el problema a uno determinístico, por ejemplo al utilizar una estimación subjetiva del escenario más probable o alguna esperanza de entre todos los escenarios, y luego resolver el problema de optimización determinista resultante. La mayor desventaja de esta metodología es la inhabilidad de reconocer la presencia palpable de instancias de datos sobre otras *"más probables"* usadas para generar la decisión óptima. Para estas instancias, las decisiones generadas pueden ser substancialmente sub-óptimas.

Actualmente las metodologías más usadas en problemas bajo incertidumbre son la *Optimización Estocástica* y la *Optimización Robusta*.

La *Optimización Estocástica* es una metodología para la modelación de problemas de optimización que involucran incertidumbre en la función objetivo o restricciones. La incertidumbre aquí es caracterizada por una distribución de probabilidad en los parámetros. Aunque la incertidumbre es rigurosamente definida, en la práctica esto puede describirse mediante algunos escenarios (posibles salidas de datos) para especificar y detallar una distribución de probabilidad asociada.

La idea fundamental detrás de la programación estocástica es el concepto de *recurso*. Recurso es la habilidad de tomar acciones correctivas después que eventos aleatorios ocurren.

De los modelos de Programación Estocástica el más estudiado es el programa lineal de dos estados. Aquí el tomador de decisiones toma una decisión en un primer estado, después del cual un evento aleatorio ocurre afectando la salida de las decisiones de primer estado. Una decisión de *recurso* puede entonces hacerse en un segundo estado que compense cualquier efecto malo que pueda haber sido experimentado como resultado de las decisiones de primer estado. Una política óptima para tal modelo, es un criterio sencillo de primer estado y una colección de decisiones de recurso (reglas de decisión) definiendo cual acción de segundo estado debería ser tomada en respuesta a cada salida aleatoria.

Ahora bien, en ese contexto es posible conocer (o estimar) la probabilidad de ocurrencia de los parámetros aleatorios y después utilizar un método de optimización estocástica para resolverlo. La metodología de optimización estocástica reconoce la presencia de múltiples instancias de datos que pueden ser potencialmente realizables en el futuro y típicamente el modelo procurará generar una decisión que optimice una medida sobre el rendimiento esperado.

La falla más importante de las metodología determinista o estocástica para abordar un problema con considerable incertidumbre en los parámetros de entrada es su incapacidad de reconocer que asociada a cada decisión hay una distribución completa de resultados dependiente de qué escenario de datos se realizó realmente y por consiguiente cualquier metodología que evalué las decisiones utilizando solamente un escenario de datos, sea el esperado o el más probable está generalmente destinado a fallar [43].

Esto es particularmente importante para decisiones de naturaleza única. El tomador de decisiones sabe que, para cualquier escenario potencialmente realizable él tiene que vivir con las consecuencias, en el desempeño del sistema, de la decisión tomada. Por lo tanto es importante el comportamiento de una decisión a través de todos los escenarios potencialmente realizables.

Por otra parte, los sistemas de decisión organizacionales evalúan las decisiones que son hechas basadas en información con incertidumbre a posteriori, como si la realización del escenario actual se hubiera conocido antes de tomar la decisión. En estas circunstancias el tomador de decisiones no solamente está interesado en cómo varía el desempeño de la decisión con las diversas realizaciones de escenarios de datos, sino también en cómo el desempeño actual del sistema bajo la decisión tomada se compara con el desempeño óptimo que pudo haber alcanzado si antes de tomar la decisión se hubiera sabido el escenario que se iba a realizar. Ni la metodología determinista, ni la estocástica capturan estos intereses.

Por lo tanto, lo que el tomador de decisión quiere no es la solución óptima para un escenario específico de datos (ni tan siquiera para el escenario más probable), ni *"óptima a largo plazo"*, sino una decisión que ejecute bien a través de todos los escenarios y se proteja contra el peor de todos. A esto se le llama decisión robusta.

Esta metodología ha sido utilizada exitosamente en diversas aplicaciones como el problema internacional de ubicación de recursos [44] o en el problema de expansión de capacidad en redes de telecomunicaciones [45], entre otras, y es la que emplearemos para dar solución al problema bajo estudio en esta tesis.

Gutiérrez, Kouvelis y Kurawarwala [28], estudian una versión no capacitada del problema de diseño de red. Ellos aplican su metodología de robustez que está basada en un criterio de *pena minmax* y adaptan la metodología de Descomposición de Benders para generar diseños robustos de red. Sin embargo, la versión capacitada del problema aquí enunciado, sobre redes no orientadas no ha sido tratado en la literatura y reviste gran importancia en problemas de telecomunicaciones los cuales están bajo constante expansión y las diferencias entre planeación a corto y largo plazo deben ser capturadas en los modelos de planeación de la red.

El problema de Diseño Robusto de Red Capacitada Multiproducto se modelará a través de un programa entero mixto no lineal (MINLP, Mixed Integer Non-linear Program, por sus siglas en inglés). Desde el punto de vista de optimización, aún en los casos más sencillos, este problema es difícil de resolver ya que se ha demostrado que pertenece a la clase *NP-completo* [19, 33, 35, 56].

Como hemos visto este tipo de problemas de diseño comprenden una gran variedad de aplicaciones. Sin embargo, en la mayoría de la literatura especializada los problemas de este tipo solo se han manejado desde el punto de vista determinista, es decir, no se contempla incertidumbre en los datos de entrada del problema. Más aún, en problemas de diseño de redes capacitadas multiproducto determinista, son muy pocos los trabajos reportados hasta la fecha sobre redes no orientadas. Esto refleja la situación en la que nos encontramos y la relevancia al incluir a este problema incertidumbre en la demanda y costos variables.

1.2 Objetivo de la Tesis

Como hemos visto el problema de optimización bajo estudio refleja una aportación interesante al considerar elementos con incertidumbre en su modelación. El

desarrollar metodologías de solución a este tipo de problemas es de gran importancia ya sea en aplicaciones de telecomunicaciones, logística, transporte u otras, donde es necesario tomar decisiones similares en situaciones que se puedan presentar sobre un horizonte de planeación a futuro.

Reconociendo la dificultad para resolver este tipo de problemas, los objetivos de esta tesis son:

- Hacer un estudio de la estructura matemática del problema de diseño de red robusto, así como de problemas semejantes a este.
- Realizar un estudio amplio de las diferentes técnicas para resolver problemas bajo incertidumbre y una extensa investigación de trabajos referentes a características aleatorias en problemas.
- Desarrollar e implementar una metodología de búsqueda para el diseño robusto de red capacitada multiproducto.
- Generar una base de instancias para este tipo de problemas que sirvan de peldaño para investigaciones futuras por parte de otros investigadores, además que los resultados de esta metodología sean utilizados como comparativo contra trabajo futuros.
- Efectuar un estudio computacional exhaustivo de la metodología de búsqueda propuesta.

1.3 Alcance del Trabajo

Con la realización del presente trabajo se provee un sistema computacional basado en la metodología propuesta para encontrar *soluciones robustas* al problema de diseño robusto de red capacitado multiproducto. El sistema que se entrega está basado en rutinas desarrolladas en lenguaje C, así mismo se hace uso del modelador de problemas de optimización GAMS 5.3 [38] utilizando el optimizador CPLEX versión 6.6 [8].

Para su ejecución se requiere de archivos con cierto formato específico para la lectura de las instancias de red, lo cual se detallará en el capítulo relacionado al diseño

computacional.

El sistema fue probado para instancias con las siguientes características, 30 y 50 nodos, 30 y 50 productos, 10 y 20 escenarios, para capacidades restringidas y holgadas, la densidad de red se manejó a un 50% y 75%.

1.4 Organización de la Tesis

Este trabajo se estructura de la siguiente forma: En el Capítulo 2 abordamos antecedentes científicos y de ayuda, así como el marco teórico para el problema de diseño robusto de red capacitado multiproducto. En el Capítulo 3 se define el modelo matemático, así como suposiciones hechas al problema en cuestión. En el Capítulo 4 detallamos los aspectos de la metodología de solución, en el Capítulo 5 se presenta el diseño del experimento computacional y en el Capítulo 6 se evalúa el desempeño de la metodología y se analizan sus resultados computacionales. Finalmente, en el Capítulo 7, se presentan conclusiones, aportaciones científicas del presente trabajo y recomendaciones para trabajos futuros.

CAPÍTULO 2
ANTECEDENTES CIENTÍFICOS Y MARCO TEÓRICO

En este capítulo presentamos algunos de los trabajos más importantes en cuanto al diseño de red determinista. También se realiza una detallada mención de trabajos en donde se ha empleado la *Optimización Robusta* (*RO,* robust optimization, por sus siglas en inglés) como metodología de solución a problemas bajo incertidumbre. Además, se presenta un marco teórico sobre la *Optimización Estocástica* y *Optimización Robusta.*

2.1 Diseños de Redes Deterministas

En 1996, Herrmann, Ioannou, Minis y Proth [30] consideraron el problema de diseño de redes capacitadas en grafos no dirigidos, y presentaron una técnica ascendente dual para encontrar buenas cotas inferiores y soluciones cercanas a la óptima. Sin embargo, por una parte sólo es aplicable a redes muy pequeñas (60 aristas y 35 productos) tipo malla y por otra, fue demostrado posteriormente que las cotas inferiores no mejoraban la relajación lineal [23].

Los investigadores Balakrishnan y Magnanti [2], presentan un procedimiento para problemas de diseño de redes a gran escala y se hace sobre redes no dirigidas, generalizando el algoritmo de ascenso dual usado para resolver los problemas de rutas más cortas, ubicación de plantas, redes de *Steiner* y árboles de expansión dirigida. Debido a que la red es no capacitada, para un diseño determinado de red, este problema se descompone en problemas de ruta más corta, un problema para cada producto que fluye en la red. Las instancias de red que probaron los autores fueron con 20 nodos, 80 arcos y 380 productos, así como con 45 nodos, 500 arcos y 1980 productos. En total, probaron 117 instancias. Los investigadores no intentaron resolver los problemas hasta optimalidad. Para casi todas las

instancias el intervalo entre las cotas superiores e inferiores del algoritmo de ascenso dual fue expresado como un porcentaje de la cota inferior, el cual, estuvo a lo más a un 4%.

El trabajo realizado por Crainic, Gendreau y Farvolden [9], sobre redes dirigidas y capacitadas, hace uso de la meta-heurística Búsqueda Tabú y el método *simplex* para encontrar soluciones relativamente buenas al problema de diseño de redes multiproducto. La contribución del trabajo es que se obtienen soluciones buenas para problemas reales de hasta 400 productos con 20 nodos y 230 arcos. Sin embargo en los problemas grandes tiene variaciones desde un 17% hasta un 32% con respecto a la mejor solución entera entregada por un optimizador después de 6 horas.

El trabajo realizado en 1998 por Holmberg y Yuan [36], propone una técnica para encontrar la solución óptima a un conjunto de modelos de diseño de redes con costos fijos. Estos modelos pueden ser capacitados o no capacitados, dirigidos o no dirigidos, incluyendo costos variables escalonados y requerimientos de supervivencia. En este trabajo se propone una heurística lagrangeana compuesta por una relajación lagrangeana, un procedimiento de optimización por subgradiente, y una heurística primal. Para probar el algoritmo analizaron un total de 65 problemas en 7 grupos de diferentes estructuras. Los rangos de nodos fueron de 7 a 150, en arcos de 42 a 1000 y en productos de 5 a 282. De los 65 problemas en 39 se presentaron mejores soluciones y en 47 la metodología propuesta fue más rápida. Las comparaciones fueron contra la mejor solución entera entregada por el optimizador CPLEX.

Dos años más tarde los mismos Holmberg y Yuan [37], implementaron un método para el diseño de redes capacitadas multiproducto donde la red es dirigida. El método de solución empleado se basa en una heurística lagrangeana insertada dentro de un contexto de *ramificación y acotamiento*. La contribución de este trabajo es poder obtener soluciones exactas, o bien soluciones aproximadas dependiendo del tiempo disponible y de la calidad de solución buscada. Se trabajó con problemas de tamaños diferentes variando el número de parámetros de la siguiente manera: nodos de 17 a 150, arcos de 272 a 1000, y productos de 16 a 282, subdivididos en 7 grupos de problemas, con un total de 58 problemas probados. Hicieron una comparación usando CPLEX para resolver los problemas después de cierto tiempo predeterminado. En 60% de los problemas analizados el método propuesto

presento mejores resultados que CPLEX.

Más recientemente en los trabajos desarrollados por De-Alba, Álvarez y González-Velarde [12, 13, 15] y De-Alba [14], se aborda un problema de diseño de red capacitado multiproducto determinístico. El problema se modela como un programa entero-mixto. En [12, 13] implementan un algoritmo basado en la metaheuristica *GRASP* para encontrar soluciones iniciales al problema de diseño de red en cuestión. Cien instancias fueron generadas aleatoriamente y separadas en 4 grupos. El criterio de comparación fue contra la mejor solución entera reportada por CPLEX, los resultados arrojados por el *GRASP* propuesto por los autores reflejan estar a un 5% del criterio de comparación. Lo más relevante que señalan ellos es el tiempo de cómputo empleado por su rutina en comparación con el optimizador CPLEX. De-Alba [14] y De-Alba, Álvarez y González-Velarde [15] diseñan una heurística basada en un marco de búsqueda dispersa (SS, Scatter Search por sus siglas en inglés), que trabaja con una población obtenida mediante un *GRASP*, diseñado por ellos mismos. Se realizó un experimento computacional para el cual fueron generadas instancias aleatorias de 30 nodos y 750 aristas con 10, 50 y 100 productos, 50 nodos y 1800 aristas con 10, 50 y 100 productos, los cuales presentan soluciones a un 3.35% respecto a la mejor solución entera arrojada por un optimizador como CPLEX. Además realizaron una comparación contra una metodología de ascenso dual [30].

En el 2004, Cobos-Zaleta [6] implementó un algoritmo basado en la metaheurística búsqueda tabú (tabú search, TS, por sus siglas en inglés) para el mismo problema. El comportamiento del método basado en TS es muy similar al basado en SS [14] para las redes con 30 nodos y 50 nodos y pocos productos. Sin embargo para las redes más grandes, donde se degrada la efectividad de SS, el método de Cobos-Zaleta presenta mejores resultados.

2.2 Manejo de la Incertidumbre

Muchos autores dividen el ambiente de la toma de decisión en tres categorías: con certeza, con riesgo y con incertidumbre [29]. En situaciones bajo certeza, todos los

parámetros son deterministas y conocidos, mientras que en las situaciones bajo riesgo e incertidumbre se involucra la aleatoriedad. Se considera que en situaciones de riesgo, existen parámetros bajo incertidumbre cuyos valores son gobernados por distribuciones de probabilidad que son conocidas por el tomador de decisiones. En situaciones bajo incertidumbre, los parámetros son inciertos, y además, en ocasiones no se tiene ninguna información sobre probabilidades. Los problemas en situaciones bajo riesgo se conocen como problemas de optimización estocástica; un objetivo común es optimizar el valor esperado de una cierta función objetivo. Los problemas bajo incertidumbre se conocen como problemas de optimización robusta y a menudo realizan la optimización del sistema bajo la premisa de que ocurrirá lo peor *(worst-case)*[61]. Observemos que los ambientes con riesgo son más generales que los ambientes con certeza y los ambientes con incertidumbre son más generales que los ambientes con riesgo.

El objetivo de la optimización robusta es encontrar una solución que *ejecute bien* bajo cualquier realización posible de los parámetros aleatorios. La definición de *ejecutar bien* varía de una aplicación a otra, y elegir una medida de funcionamiento apropiada es parte del proceso de modelación. Los parámetros aleatorios pueden ser continuos o descritos por escenarios discretos. Si se conoce o se tiene información de la probabilidad, la incertidumbre se describe usando distribuciones de probabilidad (continua o discreta) en los parámetros. Si no se conoce o se tiene información de la probabilidad, los parámetros continuos son generalmente restringidos en algunos intervalos pre-especificados.

2.2.1 Programación Estocástica

Muchos problemas de decisión pueden ser modelados utilizando programación matemática, la cual pretende maximizar o minimizar alguna función objetivo que depende de una decisión. Las posibles decisiones son restringidas por límites en recursos, requerimientos mínimos, etc. Las decisiones son representadas por variables, la cuales pueden ser por ejemplo, no negativas o enteras. Los objetivos y restricciones están en función de las variables.

La *Optimización Estocástica* es una metodología para la modelación de problemas de optimización que involucran incertidumbre en la función objetivo o restricciones.

Mientras problemas con optimización determinista son formulados con parámetros conocidos, los problemas en el mundo regularmente incluyen algunos parámetros aleatorios. La incertidumbre aquí es caracterizada por una distribución de probabilidad en los parámetros. Aunque la incertidumbre está rigurosamente definida, en la práctica esto puede detallarse mediante algunos escenarios (posibles salidas de datos) para especificar y describir una distribución de probabilidad asociada.

Cuando algunos datos son aleatorios, entonces las soluciones y el valor de la función objetivo es así mismo aleatorio. Una distribución de decisiones óptimas es generalmente no implementable.

Uno de los objetivos de la Programación Estocástica es tomar esta incertidumbre en cuenta y ofrecer decisiones óptimas a este tipo de problemas bajo aleatoriedad.

2.2.1.1 Modelo de Recurso

La idea fundamental detrás de la programación estocástica es el concepto de *recurso*. Recurso es la habilidad de tomar acciones correctivas después que ocurren los eventos aleatorios.

En el modelo de programación lineal estocástica de dos etapas el tomador de decisiones toma alguna decisión en la primer etapa, después del cual un evento aleatorio ocurre afectando la salida de las decisiones de la primera etapa. Una decisión de *recurso* puede entonces ser hecha en la segunda etapa que compense cualquier efecto malo que pueda haber sido experimentado como resultado de la decisión de la primera etapa. Una política óptima para tal modelo, es un criterio sencillo de primera etapa y una colección de decisiones de recurso (reglas de decisión) definiendo cuál acción de segunda etapa debería ser tomada en respuesta a cada salida aleatoria.

Los modelos de recurso pueden ser extendidos a un amplio número de variantes. Uno de los más comunes es incluir más etapas. Como un problema multi-etapa u horizonte finito, se toma una decisión en este momento, se espera a que alguna aleatoriedad sea descubierta, y luego se toma una decisión basada en lo que pasó. El objetivo es minimizar los costos esperados de todas las decisiones tomadas [5].

2.2.1.2 Resolver Problemas de Recurso

Resolver un problema de recurso es generalmente mucho más difícil que una versión determinista. La parte difícil es evaluar los costos esperados en cada etapa (excepto el primero).

Cuando los datos con incertidumbre están distribuidos en forma discreta, el problema puede escribirse como un problema determinista excesivamente grande. La esperanza puede ser escrita como una suma finita, y cada restricción puede ser duplicada para cada realización de datos aleatorios. El problema determinista equivalente puede ser resuelto como cualquier modelo de programación lineal, pero su estructura especial permite el empleo de técnicas de descomposición que resuelven de forma simultánea varios sub- problemas deterministas (asociados a cada escenario) rápidamente.

Cuando los datos tienen una distribución continua, la integración es particularmente difícil. Sin embargo, podemos encontrar cotas superiores e inferiores a la esperanza a los costos de recurso que reduzcan a distribuciones discretas el problema estocástico.

2.2.1.3 Formulación Estocástica

El problema que nos interesa puede formularse como uno de programación estocástica de dos etapas, esto es, las variables de decisión pueden dividirse en dos conjuntos. El primer conjunto (variables de la primera etapa) lo forman las variables de diseño, es decir aquellas relacionadas con las decisiones de qué aristas incluir en el diseño. El segundo conjunto de variables (segunda etapa) son variables de recurso y definen el flujo de cada producto que circulará por cada arista según el escenario que ocurra. El objetivo es determinar las variables de la primera etapa de tal forma que la suma de los costos de la primera etapa y el valor esperado de los costos en la segunda etapa sea mínima.

La formulación general de un problema lineal de dos etapas es:

$$\min z = c^T x + E_\xi \left[\min q(\omega)^T y(\omega) \right]$$
$$s.a. \qquad Ax = b,$$
$$T(\omega)x + Wy(\omega) = h(\omega),$$
$$x \geq 0, y(\omega) \geq 0.$$

Las decisiones de primer estado son representadas por el vector x con dimensiones $n_1 \times 1$. Correspondientemente a x son los vectores de primer estado y matrices c, b y A, dimensiones $n_1 \times 1$, $m_1 \times 1$ y $m_1 \times n_1$, respectivamente. En el segundo estado, un número de eventos aleatorios $\omega \in \Omega$ pueden ser realizados. Para cualquier realización dada de ω, los datos del problema de segundo estado $q(\omega)$, $h(\omega)$ y $T(\omega)$ llegan a ser descubiertos, donde las dimensiones son $n_2 \times 1$, $m_2 \times 1$, $m_2 \times n_1$ respectivamente.

Cada componente de q, T y h es de este modo una variable aleatoria. Permitamos que $T_i.(\omega)$ sea la i-ésima fila de $T(\omega)$. Juntando los componentes estocásticos de los datos de la segunda etapa, se obtiene un vector $\xi^T(\omega) = (q(\omega)^T, h(\omega)^T, T_1.(\omega), \ldots, T_{m2}.(\omega))$, con $N = n_2 + m_2 + (m_2 \times n_1)$ componentes. Como se indico antes, un evento aleatorio dado ω influencia a varias variables aleatorias, en este caso, todas las componentes de ξ.

Como se menciono antes, cuando un evento aleatorio ω es realizado, los datos del problema de segundo estado, q, h, y T, llegan a ser descubiertos. Entonces, las decisiones de segundo estado $y(\omega)$ o $(y(\omega,x))$ pueden ser tomadas. La función objetivo del problema lineal de dos estados contiene un término determinístico $c^T x$ y la esperanza de la función objetivo de segundo estado $q(\omega)^T y(\omega)$ tomada sobre todas las realizaciones de eventos aleatorios ω. Este término de segundo estado es el más difícil debido a cada ω, el valor $y(\omega)$ es la solución a un problema lineal. Para recalcar este hecho, algunas veces se utiliza el concepto de programa determinístico equivalente. Para una realización de ω, permitamos que

$$Q(x, \xi(\omega)) = \min_y \left\{ q(\omega)^T y \mid Wy = h(\omega) - T(\omega)x, y \geq 0 \right\}$$

sea el valor de la función de segundo estado. Entonces, se define la esperanza de la función de segundo estado

$$Q(x) = E_\xi Q(x, \xi(\omega))$$

y el programa determinístico equivalente es

$$\begin{aligned} \min\ & z = c^T x + Q(x) \\ s.a.\quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

Esta representación de un programa estocástico claramente ilustra que la mayor diferencia respecto a una formulación determinista es el valor de la función de segundo estado. Si esta función es dada, entonces el programa estocástico es solo un programa no lineal ordinario [5].

2.2.2 Optimización Robusta

Cuando no se tiene información de la probabilidad acerca de los parámetros bajo incertidumbre, el costo esperado y otros objetivos mencionados no se pueden evaluar. Muchas medidas de robustez han sido propuestas para esta situación. Las dos más comunes son el *costo minmax* y la *pena minmax*. Varias de estas medidas de robustez son discutidas en el texto de optimización robusta escrito por Kouvelis y Yu [43].

Como en el caso de la optimización estocástica, los parámetros bajo incertidumbre en problemas de optimización robusta se pueden modelar como discretos o continuos. Los parámetros discretos son modelados utilizando una metodología de escenarios. Los parámetros continuos se toman generalmente en ciertos intervalos especificados de antemano, debido a que, a menudo es imposible considerar un *"escenario de peor caso"* cuando los valores del parámetro son no acotados. Referiremos este tipo de incertidumbre como *"incertidumbre de intervalos"* y se referirán los parámetros modelados de esta misma manera como parámetros *"inciertos en intervalo"*.

Para un problema dado bajo incertidumbre sin información sobre la probabilidad, la solución de *costo minmax* es aquella que minimiza el máximo de los costos a través de todos los escenarios. Esta medida es, por una parte, excesivamente conservadora y pesimista, acentuando el peor escenario, y por otro lado, algo descuidada, debido a que puede producir soluciones bastante pobres. El *costo minmax* puede ser una medida

apropiada para situaciones en las cuales es crítico para el sistema funcionar bien aún en el peor caso.

Otras dos medidas comunes de robustez consideran la pena o arrepentimiento de una solución, la cual es la diferencia (absoluta o en porcentaje) entre el costo de una solución en un escenario dado y el costo de la solución óptima para ese escenario. La pena en algunas ocasiones es descrita como una pérdida de oportunidad: la diferencia entre la calidad de una estrategia dada y la calidad de la estrategia que se habría escogido si se tuviera conocimiento sobre la elección correcta en el futuro. Los modelos que procuran minimizar el máximo (absoluto o relativo) de la pena a través de todos los escenarios son conocidos como modelos de *pena minmax (absoluta o relativa)*. Los problemas de *costo minmax* a menudo se pueden transformar en problemas equivalentes de *pena minmax*, y viceversa, debido a que el costo y la pena de un escenario dado difieren sólo por una constante. Los enfoques de la solución para un criterio son a menudo aplicables al otro, también.

2.2.2.1 Definición Formal de Optimización Robusta

Se utiliza una metodología basada en escenarios para representar instancias con incertidumbre a un modelo dado. La *optimización robusta* depende crucialmente del proceso de generación de escenarios, así como de la agudeza e intuición del tomador de decisiones acerca del ambiente de decisión.

Definamos a S como el conjunto potencial de estados o escenarios realizables sobre un horizonte de planeación prespecificado. Sea X el conjunto de las variables de decisión y D sea el conjunto de entrada de datos. Se usa la notación D^s para indicar la instancia de datos correspondiente al escenario s. Permitamos que F_s indique el conjunto de soluciones factibles cuando el escenario s se realiza, y supongamos que el costo de la decisión $x \in F_s$ es evaluada utilizando la función $f(x, D^s)$ (observar la dependencia de la función de costo respecto a la decisión x y a la instancia de datos). Entonces, la decisión óptima x^*_s a la entrada de datos D^s es la solución a un problema de optimización determinista que satisface

$$z^s = f(x_s^*, D^s) = \min_{x \in F_s} f(x, D^s)$$

En la introducción de esta tesis mencionamos los criterios de optimización robusta, aquí los definiremos rigurosamente. La *decisión robusta absoluta* x_A se define como aquella que minimiza el máximo de los costos totales, de entre todas las decisiones factibles sobre todas las instancias de escenarios realizables, esto es

$$z_A = \max_{s \in S} f(x_A, D^s) = \min_{x \in \cap_{s \in S} F_s} \max_{s \in S} f(x, D^s)$$

Las decisiones *robustas absolutas* son de naturaleza conservadora ya que están basadas en la anticipación de que lo peor puede suceder. Ambientes que motivan a tal criterio es en situaciones de competencia, donde parámetros del modelo de decisión están afectados por las acciones de los competidores (si los intereses del competidor están en conflicto con los míos, debo de anticipar la realización de escenarios de datos que puedan evitar reducir mis ganancia al mínimo).

La *decisión de desviación robusta* x_D es definida como aquella que exhibe el mejor peor caso de desviación de la optimalidad entre todas las decisiones factibles sobre todos los escenarios de datos realizables,

$$z_D = \max_{s \in S} (f(X_D, D^s) - f(x_S^*, D^s)) = \min_{x \in \cap_{s \in S} F_s} \max_{s \in S} (f(x, D^s) - f(x_S^*, D^s))$$

La decisión robusta relativa x_R se define como aquella que exhibe el mejor peor caso de desviación porcentual de la optimalidad entre todas las decisiones factibles sobre todos los escenarios de datos realizables,

$$z_R = \max_{s \in S} \frac{f(x_R, D^s) - f(x_S^*, D^s)}{f(x_S^*, D^s)} = \min_{x \in \cap_{s \in S} F_s} \max_{s \in S} \frac{f(x, D^s) - f(x_S^*, D^s)}{f(x_S^*, D^s)}$$

Estos dos últimos criterios son apropiados para ambientes en los cuales la calidad de la decisión es evaluada a posteriori, también es apropiada para ambientes de mercado altamente competitivo donde se requiere que el desempeño de la firma sea satisfactorio (cercano al de sus competidores) en cualquier escenario. Estos criterios permiten referenciar el desempeño de la decisión contra las mejores salidas posibles en cualquier escenario realizable y por consiguiente acotar la magnitud de las oportunidades pérdidas en los diversos escenarios, lo cual pudiera ser explotado por los competidores.

2.2.2.2 Marco de la Estructura de la Toma de Decisiones Robustas

Existen tres elementos críticos en la aplicación de la *metodología robusta* para la toma de decisiones, lo cual está representado mediante una estructura de toma de decisiones sistemática de la siguiente forma:

a) Uso de una *metodología de planificación de escenarios* para estructurar la incertidumbre en los datos, para la situación de decisión.

b) Elección del (o los) *criterio (s) de robustez* apropiado a la situación de decisión.

c) El *desarrollo formal de un modelo de decisión*, el cual es una forma especial del modelo de optimización bajo estudio.

Tan pronto se junten estos tres elementos, el desarrollo de los cuales constituye la parte artística de la metodología robusta en la toma de decisiones, entonces el tomador de decisiones puede emplear la programación matemática estándar (u otras herramientas computacionales) para generar soluciones robustas.

2.3 Aplicaciones de la Optimización Robusta

Los modelos de *pena minmax* han sido empleados comúnmente en la literatura. Generalmente tales problemas son resueltos utilizando algoritmos específicos, tal es el caso de un algoritmo para problemas de *pena minmax* lineales con coeficientes objetivos inciertos en intervalo que fue propuesto por Mausser y Laguna [52, 54]. El algoritmo depende del hecho de que para una solución dada, cada parámetro incierto se fija ya sea en su nivel inferior o nivel superior en un escenario de pena maximizado. Para identificar este escenario, los autores resuelven un problema entero mixto que agrega una variable binaria y unas pocas restricciones al modelo original para cada parámetro incierto. Este enfoque es práctico para problemas lineales de tamaño pequeño a moderado.

Mausser y Laguna [53] proponen una heurística voraz para el problema de pena absoluta que contiene algunos métodos para la diversificación y para evitar óptimos locales.

La estrategia general del algoritmo para problemas de *pena minmax* puede ser descrita como sigue:

1. Elegir una solución candidata x.

2. Determinar la pena máxima a través de todos escenarios una vez que la solución x es elegida. Para escenarios discretos, esto es fácil: resolver el costo de la solución bajo cada escenario y compararlo contra la solución óptima de cada escenario, después elegir el escenario con la pena mayor. Para la incertidumbre en intervalo, las técnicas para encontrar el escenario de pena máximo dependen del hecho que este escenario tiene típicamente cada parámetro fijo al punto final de su intervalo. Aún el problema de identificar este escenario puede ser bastante difícil. Resolver este problema es el punto más importante de los algoritmos propuestos por Mausser y Laguna [52, 53, 54].

3. Ya sea repetir los pasos 1 y 2 para todas las posibles soluciones como en [1], o encontrar de algún modo una nueva solución candidata cuya pena sea más pequeña que la determinada en el paso 2 como en Mausser y Laguna [52, 54].

Antes que minimizar la pena, varios trabajos han involucrado restricciones en la pena máxima que pueda ser alcanzada por la solución. Esta idea fue primero utilizada por Kouvelis, Kurawarwala y Gutiérrez [42], quienes imponen una restricción que dicta que la pena relativa en cualquier escenario no debe ser mayor que p, donde $p \geq 0$ es un parámetro externo. Es decir, el costo bajo cada escenario debe estar dentro de $100(1+p)$ % del costo óptimo para ese escenario. En el 2003, Snyder y Daskin [62] se refieren a esta medida como *"p-robustez"*. Para p pequeñas, es posible que no haya soluciones *p-robustas* para un problema dado, así que, la *p-robustez* agrega un aspecto de factibilidad no presente en otras medidas de robustez. El problema considerado por Kouvelis, Kurawarwala y Gutiérrez [42] es un problema de diseño de instalaciones en el cual el objetivo es construir una lista de soluciones *p-robustas*, si existe alguna. La metodología usada es una heurística en cierto sentido, aunque se resuelva hasta optimalidad el problema de diseño de instalaciones en

cada escenario, no hay garantía que la lista resultante de soluciones *p-robusta* sea exhaustiva.

El criterio de *p-robustez* es utilizado también por Gutiérrez y Kouvelis [27] en el contexto de un problema internacional de suministro. Ellos presentan un algoritmo que para una p y N dados, retorna todas las soluciones *p-robustas* (si hay menos que N de ellas) o las N soluciones con la pena máxima más pequeña.

En el trabajo desarrollado por Mulvey, Vanderbei, Zenios [57] se describen aspectos de optimización robusta, y aplicaciones de esta metodología a casos concretos. En este trabajo se describen provechosas propiedades de la solución a un modelo cuando los datos del problema son descritos por un conjunto de escenarios, en lugar de puntos de estimación. Aquí es donde se introduce la siguiente terminología ya común actualmente: Una solución a un modelo de optimización es definida como *solución robusta* si ésta permanece "cerca" del óptimo para todos los escenarios definidos, y como *modelo robusta* si ésta permanece "casi" factible para todos los escenarios definidos. Hacen el desarrollo de una formulación general, la cual denominan *optimización robusta (RO),* además hacen una comparación entre *RO* contra metodologías tradicionales como análisis de sensibilidad y programación lineal estocástica.

En el modelo robusto descrito por Mulvey [57] se definen dos componentes, una componente estructural que es fija y libre de incertidumbre, así como una componente de control que es sujeta a incertidumbre. Una novedad que introducen es el uso de momentos máximos de la distribución de la función objetivo en el modelo de optimización. Introducen una función de utilidad que involucra un marco entre el valor medio y la variabilidad de este valor medio. Además el uso de momentos máximos en este modelo distingue las características de *RO* de la optimización lineal estocástica. Un segundo término en la función objetivo es utilizado y es considerado un término de penalidad por factibilidad. Desarrollan la metodología para varias aplicaciones del mundo real como problemas de expansión de capacidad de energía, balance de una matriz y reconstrucción de imagen, programación de una línea aérea, planeación financiera y un diseño estructural de mínimo peso. Esta metodología ha sido utilizada exitosamente en diversas aplicaciones como en el

problema de ubicación internacional de recursos [44], o en el problema de expansión de capacidad en redes de telecomunicaciones [45].

De la metodología robusta introducida por Kouvelis y Yu [43] varios trabajos han sido reportados considerando las estrategias descritas, por ejemplo para problemas lineales [53] y para el problema de diseño de redes [31].

En otro estudio, Eppen, R. Kipp-Martin y Shrage en 1988 [18] modelan el problema de planeación estratégica de la capacidad de un fabricante de automóviles importante. Las decisiones en el modelo incluyen la creación o cierre de instalaciones para algunas de las líneas de productos bajo consideración. La fuente principal de incertidumbre en el modelo es la demanda del producto sobre un período de planeación a mediano plazo.

Laguna [45] trabaja con un problema en una compañía de telecomunicaciones. En este problema la demanda para la capacidad de la red es originada por movimientos de clientes y la introducción de nuevos productos y servicios. La planeación tradicional de capacidad da inicio con un pronóstico de la demanda basada en información demográfica. Asimismo, las decisiones actuales de expansión de la capacidad no pueden ser basadas en técnicas tradicionales de pronóstico (tal como regresión y modelos de series de tiempo), debido a que éstas no satisfacen o predicen la demanda para nuevos servicios. En el estudio de este problema se utiliza la metodología de *RO* para modelar la incertidumbre en el problema sobre un horizonte de tiempo a futuro. La principal idea en el trabajo es definir una colección de representaciones del modelo como un conjunto de escenarios. El resultado es un problema de gran escala incrementado en la función objetivo original por introducir nuevos objetivos que garanticen que las recomendaciones del modelo sean cercanas al óptimo, independientemente de que escenario ocurra. Cuando esta situación se origina, se dice que la solución es robusta sobre el universo de escenarios. El problema es formulado mediante el marco de optimización robusta y resuelto exactamente en dos fases. La primera fase consiste en programación dinámica recursiva y la segunda en un procedimiento de la ruta más corta.

Laguna y González-Velarde [44] trabajaron con el problema de ubicación internacional de recursos. La principal contribución de este trabajo es suponer capacidad finita, lo opuesto a la suposición de capacitad infinita manejada en estudios anteriores. La

formulación también trata incertidumbre en parámetros importantes del problema tal como la demanda y tasas de cambio, además se utiliza una función objetivo que incorpora una medida de riesgo. Como se sabe, el problema de ubicación internacional de recursos consiste en seleccionar un subconjunto de proveedores de un conjunto de proveedores internacionalmente ubicados. Para la selección de los proveedores debe conocerse la demanda de artículos de un conjunto de plantas, las cuales son también ubicadas internacionalmente. El proceso de selección involucra la optimización de una función objetivo que podría ser un costo o una medida de ganancia. Modelan la incertidumbre vía escenarios y asignan la probabilidad de ocurrencia a cada uno de los escenarios. No utiliza los costos esperados como un único término en la función objetivo a optimizar, en adición a ello utilizan un segundo término en la función objetivo para medir la variabilidad de los costos asociados con la selección de un proveedor en particular. El método de solución de este problema puede ser visto como una heurística basada en el paradigma de Descomposición de Benders. Los investigadores aprovechan la información sobre los valores duales asociados a una de las restricciones en el problema para determinar el grado de importancia de un nodo, y de esta forma tomar decisiones de movimientos sobre ellos (eliminación e inserción de nodos). Esta metodología juega un importante rol dentro del desarrollo de este trabajo de tesis, ya que de este trabajo se desprenden ingeniosas estrategias para resolver el problema que nos atañe.

En el trabajo desarrollado por Raja y Han [59], se investiga el diseño óptimo para redes de acceso local (LACN local access network por sus siglas en inglés) derivado de resolver un problema de ubicación de concentradores capacitados (CCLP capacitated concentrator location problem por sus siglas en inglés) con incertidumbre en el tráfico de la demanda y en costos. Ahora bien, la problemática consiste en una red de acceso local (LACN) que es establecida conectando los nodos del usuario final (ejemplos, PC, Workstation (sitios de trabajo), LANs) a dispositivos multiplexicos (concentradores) desplegados en determinados puntos vía enlaces de banda baja. Además cada concentrador en la LACN es conectado más internamente con un nodo WAN (wavelength access network) vía un enlace de banda ancha. El problema de ubicación de concentradores capacitados (CCLP) se utiliza típicamente para caracterizar diseños de LACN y la interconexión de LACN-WAN. El objetivo de CCLP es seleccionar el número óptimo de

concentradores en los sitios predeterminados tales que cada nodo del usuario final esté conectado con exactamente un concentrador, sin violar la capacidad de ningún concentrador, mientras se minimiza el costo total de la red. En una estrategia pionera, este trabajo soluciona una versión no determinista de CCLP usando la metodología de *RO*. La *RO* fue satisfactoria para los problemas CCLP que implicaban incertidumbre en parámetros de diseño. La metodología de *RO* utilizada en este trabajo se muestra atractiva ya que no requiere conocimiento específico de distribuciones de probabilidad para caracterizar la incertidumbre en los parámetros del diseño de red.

Este trabajo presenta una aplicación pionera de la optimización robusta basada en escenarios (*SRO*, Scenario-based Robust Optimization por siglas en inglés) para el diseño de red de acceso local, la cual agrega una nueva dimensión al diseño de redes y planeación de instalaciones. En particular, este trabajo desarrolló un modelo matemático y un algoritmo para resolver el problema CCLP sujeto a incertidumbre en la demanda de banda y costos.

El modelo *SRO* que se desarrolló en este trabajo caracterizó la incertidumbre vía dos escenarios discretos alternativos. Cada valor del escenario corresponde a un caso de datos de entrada que puede ser realizado con una probabilidad positiva no necesariamente conocida. El algoritmo de solución se conduce por un proceso paralelo de ramificación y acotamiento (*branch and bound)* para la búsqueda de soluciones robustas para los dos escenarios dados. El algoritmo fue probado al solucionar 60 problemas generados aleatoriamente con cuatro diversos tamaños del problema. El algoritmo pudo encontrar por lo menos una o más soluciones robustas para 52 de los 60 problemas probados. Las soluciones producidas por *SRO* representan un diseño de red mejor *(robusto)* que es insensible a los cambios potenciales en los costos de comunicación y demanda de banda, además en la mayoría de los casos, las soluciones fueron factibles y cercanas del óptimo estando dentro del 3% a través de todos los escenarios definidos.

En 1999, Herrmann [31] propuso un algoritmo genético de dos fases como una técnica general para resolver problemas de optimización *minmax*. El utilizó la optimización robusta discreta como técnica para estructurar la incertidumbre en el proceso de toma de decisiones. El objetivo en este problema es encontrar una solución robusta que tenga el

mejor rendimiento en el problema del *"peor caso"* sobre un conjunto dado de escenarios. El algoritmo genético que propone el autor para este tipo de problemas mantiene dos poblaciones. La primera población representa las soluciones. La segunda población representa los escenarios. Un individuo en una población es evaluado con respecto a los individuos en otra población. Las poblaciones evolucionan simultáneamente, y convergen a una solución robusta y al escenario del *"peor caso"*.

En el 2003, J. Kennington, K. Lewis, E. Olinick, A. Ortynsky y G. Spiride [39] trabajaron con el problema de ruteo y aprovisionamiento de WDM (wavelength division multiplexing, por sus siglas en inglés). El estudio trata sobre redes ópticas WDM consideradas como candidatas primarias para el transporte de datos en el futuro, ya que el proceso simplificado y la administración de estas redes de banda alta las hacen muy atractivas. Se anticipa que estas redes serán utilizadas para transacciones comerciales vitales tales como órdenes de proceso y control de inventario *(e-Commerce)* así como para comunicación de voz. Muchas compañías de telecomunicaciones están implicadas en el diseño, desarrollo, y despliegue de estas redes. Para satisfacer la demanda del cliente en servicios informativos con una confiabilidad alta y tiempos de reacción rápidos, las redes WDM deben ser tolerantes a las fallas. Para ser económicamente viables, estas deben ser diseñadas eficientemente. El objetivo del programa de investigación de este trabajo fue desarrollar la metodología y software prototipo para el diseño y aprovisionamiento de redes WDM.

Diseñar al menor costo e involucrar fallas tolerantes en redes WDM para satisfacer un conjunto dado de puntos de demanda entre clientes fue desafiante para los autores. Los parámetros de diseño típicamente incluyen la ubicación de los enlaces físicos entre las ubicaciones de los clientes en la red, el número de las *fibras* que se utilizarán en cada enlace, el número de *canales* que se utilizarán en cada fibra, el número de *amplificadores ópticos* y de *regeneradores* necesarios para apoyar los canales, así como la localización de las *líneas terminales de equipo* requeridas para apoyar las demandas entre puntos. Además de determinar la infraestructura física de la red, un diseño completo debe ser especificado para anticipar el tráfico. En algunas aplicaciones, el usuario requiere capacidad de repuesto en la red aunada con una estrategia de protección para salvaguardar la red contra fallas en los nodos o enlaces.

Dado los costos del equipo mencionado arriba, un modelo de optimización fue construido por Kennington et al. [39] para ayudar a la selección de un diseño a costo mínimo para cierto pronóstico de la demanda. Puesto que el crecimiento en esta industria es difícil de predecir para los autores existió la preocupación que un diseño óptimo de red para un pronóstico dado fuera erróneo o inferior. Si el pronóstico es demasiado bajo, la red no tendría suficiente capacidad de manejar la demanda y el usuario de la red perdería probablemente información. Inversamente, si el pronóstico es demasiado alto el resultado se observaría en los costos de equipo poco utilizado. Los investigadores desarrollaron una metodología de diseño que considera la incertidumbre en los pronósticos de la demanda. Denominaron a su estrategia una metodología de *diseño robusto* para el aprovisionamiento de redes ópticas. Un *diseño robusto* para los autores es uno que trabaja bien sobre una amplia gama de escenarios posibles de la demanda. Utilizando su modelo matemático y sistema computacional de solución desarrollado por ellos, se encontraron soluciones para sus instancias de prueba dentro de un 5% de optimalidad.

Como podemos apreciar el marco de aplicación de *RO* se ha acrecentado en la actualidad y la mayoría de los investigadores se han inclinado por el uso de esta metodología por diversas razones. Otros trabajos donde se ha aplicado el marco de *RO* es en programación de máquinas en paralelo con interrupciones estocásticas [46], en la re-ubicación de especies animales bajo incertidumbre en el crecimiento demográfico futuro [29], en planeación de la producción [63], en sistemas logísticos de gran escala [64], en ingeniería química [10] y en ubicación de instalaciones [40].

CAPÍTULO 3
PLANTEAMIENTO DEL PROBLEMA

3.1 Introducción

En este capítulo se presenta el modelo matemático del problema de diseño de red robusto capacitado multiproducto así como algunos supuestos hechos sobre el mismo. Características, retos y dificultades también son mostrados con el propósito de ampliar a detalle la problemática que nos atañe.

3.2 Conjuntos y Parámetros

Sea $G=(N,A)$ un grafo que representa una red no orientada con un conjunto N de nodos y un conjunto A de aristas potenciales. A cada arista $\{i,j\}$ corresponden dos arcos orientados (i,j) y (j,i). Al conjunto de arcos orientados lo llamaremos A'. Sea K el conjunto tipos de productos. Del conjunto de nodos se distinguirán varias parejas origen-destino, asociadas cada una de ellas a un producto que circulará por la red. Sean $O(k)$ y $D(k)$ el nodo origen y nodo destino respectivamente del k-ésimo producto. Cada arista $\{i,j\}$ tiene asociada una capacidad finita u_{ij}, la cual deberá ser compartida por todos los productos que circulen en cualquier dirección de la misma. Por otra parte, a cada arista potencial $\{i,j\}$ se le asigna un costo fijo F_{ij} por su construcción.

Se considera incertidumbre en dos de los parámetros del problema: costos de ruteo y demandas. La incertidumbre como ya se había mencionado será modelada vía escenarios discretos. Sea S el conjunto de escenarios con probabilidad de ocurrencia P_s. Consideremos costo $c_{ij}^k(s)$ y $c_{ji}^k(s)$ por unidad de producto del tipo k transportado en el arco (i,j) y (j,i) respectivamente, en el escenario s. Como puede verse, estos costos dependen del tipo de producto que se trate, del sentido en que circule por la arista y del escenario que se

presente. Para cada producto $k \in K$, denotemos por $d^k(s)$ la demanda de dicho producto en el escenario s.

3.3 Suposiciones

El establecimiento de suposiciones tiene que ver con las características propias del problema planteado, y son expuestas con el propósito de señalar diferencias respecto a otro tipo de diseño de redes. Las suposiciones que son consideradas para nuestro estudio son las siguientes:

S1. *La red es no orientada.* Una vez que alguna arista $\{i,j\}$ sea elegida para estar dentro de nuestro diseño de red se considerarán los dos arcos orientados (i,j), (j,i).

S2. *La red es capacitada.* Esta característica indica que tenemos cotas de capacidad por cada arista $\{i,j\}$ declarada en el problema, y será compartida por los dos arcos asociados a cada arista y por todos los productos.

S3. *Uso de escenarios.* El uso de escenarios tiene que ver con la estrategia para representar la incertidumbre del problema de diseño de red robusto. La incertidumbre se contempla en dos de los principales parámetros de entrada, los cuales son las demandas y costos de ruteo. La incertidumbre en estos parámetros será modelada a través de escenarios. Un escenario de datos de entrada específico representa una realización potencial de parámetros importantes del modelo de decisión (es decir, ocurre con alguna probabilidad positiva, no necesariamente conocida).

En general, el uso de escenarios para estructurar la incertidumbre en los datos de entrada, permite al tomador de decisiones describir la relación de unos pocos factores principales inciertos en el ambiente de decisión con un conjunto grande de parámetros de entrada, con muchos de ellos siendo simultáneamente afectados por unos o más de esos factores. La correlación entre los factores principales que simultáneamente afectan los datos de entrada puede ser acomodada fácilmente.

S4. *La red es balanceada.* Al indicar que la red es balanceada decimos que las demandas para cada producto declarado en el problema son transportadas en su

totalidad de su nodo origen $O(k)$ a su nodo destino $D(k)$, la distribución del flujo se realiza a través de las aristas elegidas a pertenecer en la red, las demandas son satisfechas sin dejar en su trayecto por los nodos de paso alguna cantidad de flujo excedente o pérdida.

3.4 Modelo Matemático

El modelo tiene dos tipos de variables de decisión. El primer tipo es una variable binaria que modela las elecciones de diseño y se define como y_{ij}= 1, si la arista $\{i,j\}$ se incluye en el diseño de la red, o bien, y_{ij} = 0, en caso contrario. El segundo tipo es una variable continua que modela las decisiones de flujo del producto k que circula por el arco orientado (i,j) en el escenario s, y que denotaremos como $x_{ij}^{k}(s)$.

Las primeras son variables binarias y en la terminología de optimización robusta se denominan variables de diseño, esto es, ellas modelarán las decisiones que deben hacerse antes de resolver la incertidumbre. Las segundas son variables no negativas y se denominan variables de decisión de control, cuyos valores tienen que ser ajustados de acuerdo a la realización de cada escenario.

Para cada nodo i, para cada producto k y para cada escenario s impondremos las restricciones usuales de conservación de flujo en redes

$$\sum_{\{j:(i,j)\in A'\}} x_{ij}^{k}(s) \; - \sum_{\{j:(j,i)\in A'\}} x_{ji}^{k}(s) = \begin{cases} d^{k}(s) & \text{si } i = O(k) \\ -d^{k}(s) & \text{si } i = D(k) \\ 0 & \text{de otro modo,} \end{cases} \qquad (1)$$

para $k\in K$, $i\in N$ y $s\in S$.

También consideraremos que el flujo de todos los productos que circulan en cualquier dirección por cada arista $\{i,j\}$ en cada escenario s no debe exceder la capacidad de dicha arista, es decir

$$\sum_{k\in K} (x_{ij}^{k}(s) \; + \; x_{ji}^{k}(s)) \le u_{ij} y_{ij} \qquad \{i,j\} \in A, s \in S. \qquad (2)$$

Estas restricciones no sólo aseguran que sean respetadas las capacidades de las aristas, sino también fuerzan a que el flujo de cualquier producto $x_{ij}^{k}(s)$ sea cero si la arista $\{i,j\}$ no ha sido seleccionada dentro del diseño.

Por último, exigimos que las variables continuas sean no negativas y que las variables de diseño sean binarias.

$$x_{ij}^{k}(s) \geq 0 \qquad k \in K, (i,j) \in A', s \in S. \tag{3}$$
$$y_{ij} \in \{0,1\} \qquad \{i,j\} \in A.$$

Nuestro objetivo será encontrar el diseño que minimice el máximo de los costos totales (por diseño y operación de la red), de entre todas las decisiones factibles sobre todos los escenarios realizables, esto es,

$$\min_{y \in Y} \max_{s \in S} \min_{x \in X} \left[\sum_{k \in K} \sum_{(i,j) \in A'} c_{ij}^{k}(s) x_{ij}^{k}(s) + \sum_{\{i,j\} \in A} F_{ij} y_{ij} \right] \tag{4}$$

donde $X = \{(x_{ij}^{k}):\ (i,j) \in A', k \in K\}$, y $Y = \{(y_{ij}):\ \{i,j\} \in A\}$.

Al problema (1)-(4) lo referiremos como $DRCM_{UU}$. Como podemos observar el problema anteriormente definido es entero mixto no lineal (MINLP, Mixed Integer Non linear Program, por sus siglas en inglés).

3.5 Características del Problema

- Es importante distinguir en el modelo que una vez que se haya decidido conectar dos nodos i, j, la arista $\{i,j\}$ formará parte del diseño de la red, por lo que se permitirá flujo en ambos sentidos, o sea, se considerarán en la red ambos arcos (i,j) y (j,i).
- Una vez que una arista ha sido incluida en el diseño de la red, el costo de transportación en que se incurre dependerá de los productos que estén circulando por la arista, la orientación en que lo hacen y el escenario que se esté realizando.
- Obsérvese que para un arreglo de valores de las variables de diseño (y_{ij}), la

formulación del problema de diseño para un escenario dado se convierte en un problema de flujo a costo mínimo en redes multiproducto con capacidades en las aristas, y este hecho es un punto relevante que aprovecharemos para generar información que nos guíe a una búsqueda de soluciones robustas.

3.6 Retos y Dificultades

Una primera dificultad del problema original radica en la relación existente entre costos fijos y costos variables al construir una solución, así como la relación entre la capacidad finita de las aristas y los costos fijos de las mismas aristas. Ahora bien el aspecto de densidad de red puede verse reflejado en dos posibles hechos con respecto a los costos incurridos (construcción y operación): incluir muchas aristas puede aumentar la componente de costo de construcción, pero se abarataría el costo por operación, incluir menos aristas hará disminuir el costo pagado por construcción pero posiblemente incrementarán los costos de operación.

Debido a todas las posibilidades de diseños de red que tenemos en un problema de estas características este es considerado un problema combinatorio, esto refleja una dificultad al momento de encontrar por lo menos una solución inicial.

Aunado a esto tenemos el aspecto de aleatoriedad que incrementa enormemente la dimensión del problema y por lo tanto el espacio de exploración de soluciones para dicho problema es mucho más complejo. Estos aspectos elevan enormemente la dificultad cuando se intenta resolver instancias de tamaño real.

Más específicamente detallando el párrafo anterior, la dimensión de un problema de red determinístico está dado por el número de restricciones y variables del problema, ayudándonos con las restricciones (1), (2) y variables en (3), si el problema fuera determinístico, tendríamos $|K| \cdot |N| + |A|$ restricciones y $|K| \cdot |A'| + |A|$ variables. Por lo tanto, para un problema bajo incertidumbre la dimensión crece en relación al número de escenarios que se estén considerando, de tal forma que el número de restricciones (1) y (2) y variables (3) son $|S| \cdot (|K| \cdot |N| + |A|)$ y $|K| \cdot |A'| \cdot |S| + |A|$, respectivamente. Esto sin duda refleja la magnitud de los problemas con los que estamos tratando y más aun la

dificultad que representa resolverlos.

Algunas de las metodologías de solución a problemas de optimización utilizan para resolver problemas con estas características una solución inicial o conjunto de soluciones iniciales que van mejorando iterativamente. Hay que hacer mención que para $DRCM_{UU}$ el obtener una solución inicial implica un reto importante, ya que se necesita una solución, o sea, un diseño, que sea factible en cualquier escenario potencialmente realizable.

Por mencionar un ejemplo, para algunas instancias determinísticas de diseño de red de mediano tamaño (30 nodos, 350 aristas y 50 productos), luego de 8 horas, CPLEX no pudo encontrar siquiera una solución factible al problema de diseño de red determinístico, que es equivalente a considerar un solo escenario de datos.

De allí la necesidad de diseñar metodologías que ataquen de manera eficiente este tipo de problemas bajo aleatoriedad, que si bien no garantizan encontrar una solución exacta u óptima, ofrezcan una solución suficientemente buena (calidad aceptable) para tamaño reales de instancias en un tiempo razonable de forma relativamente sencilla y rápida.

CAPÍTULO 4
METODOLOGÍA DE SOLUCIÓN A $DRCM_{UU}$

4.1 Introducción

El problema de diseño de red determinístico usualmente es modelado como un problema de optimización combinatoria y se ha demostrado que es *NP-completo* [19, 33, 35, 56]. Por ello muchas de las técnicas propuestas para resolver diferentes aplicaciones del mismo toman en cuenta esta característica.

El problema de estudio en esta tesis, $DRCM_{UU}$, incorpora incertidumbre en las demandas de los productos y en los costos variables o de ruteo, la cual ha sido modelada a través de escenarios. Cada uno de los subproblemas de $DRCM_{UU}$ (asociados a los escenarios) es en sí un problema de diseño de red determinista y por tanto *NP-completo*. Esto refleja la complejidad del problema que se pretende resolver.

La metodología de solución a $DRCM_{UU}$ que en esta tesis se propone utiliza ciertas estrategias de la técnica de descomposición de Benders (BD, Benders Decomposition, por sus siglas en inglés) [4] y de la metaheurística Búsqueda Tabú (TS, Tabu Search, por sus siglas en inglés) [26]. Por lo tanto, en los puntos siguientes daremos una breve descripción de las heurísticas y metaheurísticas en general, así como BD y TS.

4.2 Heurísticas

El término heurística proviene de la palabra griega *heuriskein* relacionado con el concepto de encontrar. Se califica de heurístico a un procedimiento que encuentra soluciones de buena calidad con un costo computacional razonable, aunque no se garantice

su optimalidad. Las heurísticas son métodos para intentar resolver problemas complejos en forma aproximada, siendo una ventaja importante que su complejidad es reducida en comparación con los métodos exactos, por lo que suelen ser más fáciles de entender.

Recientemente el interés por los métodos heurísticos se ha visto incrementado considerablemente en virtud de la necesidad de dar respuesta a situaciones para las cuales es preciso ofrecer algún tipo de solución. Estas problemáticas, al ser modeladas matemáticamente son sumamente difíciles de resolver, o bien, prácticamente imposible de hacerlo. Los métodos heurísticos ofrecen una vía rápida, no de encontrar un óptimo, pero sí de encontrar soluciones de calidad aceptable a dichas problemáticas.

Los problemas de optimización combinatoria como el que estamos tratando en particular son de gran dificultad debido a su complejidad. Una característica es que el tiempo de cómputo empleado por un método exacto crece exponencialmente con el tamaño del problema, por lo cual han sido terreno fértil para la aplicación de heurísticas [16].

Existen diferentes tipos de heurísticas, las cuales pueden clasificarse como:

a) Métodos constructivos: Son aquellos que añaden componentes individuales a una solución parcial hasta que se obtiene una solución factible. El más popular de estos métodos lo constituye un algoritmo goloso o voraz "greedy", el cual construye la solución buscando el máximo beneficio en cada paso, un ejemplo es el GRASP y una aplicación en el problema de diseño de red determinístico puede verse en el trabajo de De Alba, Álvarez, González-Velarde [12, 13].

b) Métodos de descomposición: Consisten en dividir el problema en sub-problemas más pequeños, siendo la salida de uno la entrada de otro, de forma que al resolver ambos sub-problemas obtengamos una solución para el problema global. Un ejemplo de aplicación en un problema de programación lineal entera mixta, consistiría en decidir de alguna forma la solución para las variables enteras para luego resolver el problema como un LP.

c) Métodos de reducción: Identifican alguna característica que deba poseer la solución óptima y de este modo simplifican el problema. Por ejemplo la detección de alguna variable con ciertos valores o correlación, etc.

d) Manipulación del modelo: Modifican la estructura del modelo con el fin de hacerlo más sencillo de resolver, deduciendo, a partir de la solución del problema modificado, la solución del problema original. Por ejemplo se puede reducir el espacio de soluciones mediante la adición de restricciones.

e) Métodos de búsqueda por entornos: Parten de una solución factible inicial (probablemente obtenida de otra heurística) y mediante alteraciones de la solución, van iterando a otras factibles de su entorno, almacenado la mejor solución encontrada hasta que se cumpla un determinado criterio de parada.

4.2.1 Metaheurísticas

Las *metaheurísticas* son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos muy generales con un alto rendimiento. El término metaheurísticas se obtiene de anteponer a heurística el prefijo *meta* que significa "más allá" o "a un nivel superior".

En la actualidad, el principal interés por parte de los investigadores es el de diseñar métodos generales que sirvan para resolver clases o categorías de problemas. Dado que estos métodos generales sirven para construir o guiar el diseño de métodos que resuelvan problemas específicos se les ha dado el nombre de Metaheurísticos. La siguiente definición fue introducida por Osman y Nelly [58]: "Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los procedimientos metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de: inteligencia artificial, evolución biológica y mecanismos estadísticos".

Los metaheurísticos más utilizados y reconocidos en la Optimización Combinatoria son: Recocido Simulado, Búsqueda Tabú, GRASP, Búsqueda Dispersa y Algoritmos Genéticos.

4.3 Búsqueda Tabú

La Búsqueda Tabú (TS, Tabu Search, por sus siglas en inglés) es una técnica para resolver problemas combinatorios de gran dificultad que está basada en principios generales de Inteligencia Artificial (IA). Tiene sus antecedentes en métodos diseñados para cruzar fronteras de factibilidad u optimalidad local tratadas como barreras en métodos clásicos para permitir la exploración de regiones no consideradas en otro caso. El nombre y la metodología fueron introducidos por Fred Glover [25], quién la define como la guía de un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local.

TS es una técnica o procedimiento metaheurístico usado para resolver problemas de optimización combinatoria, utilizado para guiar cualquier procedimiento de búsqueda local para explorar el espacio de soluciones más allá de la simple optimalidad local.

El procedimiento local es una búsqueda cuya operación (llamada movimiento, usada para definir la vecindad de cualquier solución dada) pueda caracterizarse como una sucesión de movimientos que llevan al procedimiento de un estado (solución) a otro. El procedimiento examina soluciones "vecinas" y se mueve o avanza de forma inteligente, hacia uno de estos vecinos que provea una mejora en la solución objetivo. Este procedimiento se repite hasta que ya no sea posible mejorar una solución con respecto a las soluciones vecinas, entonces decimos que el procedimiento ha encontrado un óptimo local.

TS puede ser utilizado para guiar cualquier procedimiento de búsqueda local en la búsqueda agresiva del óptimo del problema. Por agresiva nos referimos a la estrategia de evitar que la búsqueda quede "atrapada" en un óptimo local que no sea global. A tal efecto, TS toma de la Inteligencia Artificial el concepto de memoria y lo implementa mediante estructuras simples con el objetivo de dirigir la búsqueda teniendo en cuenta la historia de ésta. Es decir, el procedimiento trata de extraer información de lo sucedido y actuar en consecuencia. En este sentido puede decirse que hay un cierto aprendizaje y que la búsqueda es inteligente.

El éxito de esta metodología en diversos problemas se debe a sus estructuras de

memoria y al uso de estrategias de intensificación y diversificación. Las estructuras de memoria evitan retornar a soluciones visitadas anteriormente, permiten guardar atributos de buenas soluciones contribuyendo a identificar regiones de interés y más generalmente guiar la exploración del espacio de solución. Las estrategias de intensificación y diversificación permiten avanzar a una solución vecina que es peor que la solución actual, pero que proporciona la posibilidad de penetrar a un espacio del conjunto de soluciones factibles que de otro modo no habría sido visitado y que podría contener una solución óptima global al problema.

TS considera dos tipos de memoria que interactúan entre sí, aunque en horizontes diferentes, una a corto plazo y otra a largo plazo, pudiendo ser éstas del tipo de frecuencia o recencia. La memoria a largo plazo tiene dos componentes que son las estrategias de intensificación y diversificación, aplicadas dentro de un ámbito global o local. El objetivo es lograr el óptimo global, evitando con ello que el algoritmo se estanque en un óptimo local.

TS se basa en la premisa de que para poder calificar de inteligente la resolución de un problema, debe incorporar memoria adaptativa (es decir, vecindades no estáticas) y exploración sensible o responsiva, que son las características principales de la búsqueda tabú. La memoria adaptativa (MA) nos permite recordar selectivamente elementos fundamentales del camino atravesado, lo cual crea un comportamiento más flexible de la búsqueda y permite la implementación de procedimientos que sean capaces de explorar el espacio de soluciones efectiva y económicamente. La exploración sensible (ES) nos permite hacer elecciones estratégicas a través del camino. El énfasis se deriva de la suposición de que una mala elección estratégica puede producir más información que una buena elección al azar. ES integra principios básicos de búsqueda inteligente, por ejemplo la exploración de características de buenas soluciones mientras se exploran nuevas regiones prometedoras.

Para guiar el procedimiento de búsqueda, TS se enfoca en imponer restricciones para negociar regiones que serían difíciles de explorar de otra manera. Estas restricciones operan en dos formas, mediante la exclusión directa de alternativas de búsqueda y mediante traslación a evaluaciones modificadas y probabilidad de selección.

Algunas aplicaciones de TS pueden verse en localización de plantas, diseño de redes de transporte, ruteo de vehículos, por mencionar algunos. Una amplia compilación de trabajos pueden ser encontradas en Díaz et al. [16].

4.4 Descomposición de Benders

El método de descomposición de Benders [4] fue publicado en 1962 y fue inicialmente desarrollado para resolver problemas de programación entera mixta. El éxito computacional del método para resolver problemas de diseño a gran escala en sistemas de distribución fue confirmado por el artículo pionero de Geoffrion y Graves [24].

Benders desarrolló una metodología inteligente para explotar la estructura de problemas de programación matemática con *variables complicadas* (variables que son temporalmente fijadas para hacer el problema restante tratable). Para una clase de problemas específicamente considerados por Benders, al fijar las variables complicadas estas reducen el problema dado a un problema ordinario de programación lineal, parametrizado, por supuesto, por el valor del vector de las variables complicadas. El algoritmo que él propuso para encontrar el valor óptimo de este vector emplea una metodología de planos cortantes para construir representaciones adecuadas de:

(i) Los puntos extremos del programa lineal como una función del vector de parametrización.

(ii) El conjunto de valores del vector de parametrización para los cuales el programa lineal es factible.

La teoría de dualidad de programación lineal fue empleada para derivar las familias de cortes naturales para caracterizar esas representaciones.

BD ha sido exitosamente utilizada en diferentes problemáticas y se puede tomar ventaja de las estructuras de una variedad de problemas de optimización tales como diseño de redes, planeación de fuentes de alimentación y optimización estocástica.

Como se había introducido, BD es un procedimiento de descomposición de variables usado para solucionar problemas complejos de programación matemática, ahora

bien, en cada iteración de este algoritmo, se soluciona un problema maestro al fijar valores a un subconjunto de variables, y un problema auxiliar completa esta asignación que produce una dualidad basada en *cortes de Benders* que son agregados al problema maestro.

BD puede ser aplicado a problemas P que tienen la siguiente estructura:

$$\begin{aligned} &\text{P: min} && f(x)+cy \\ &\text{Sujeto a:} && Ax+Gy \geq b \qquad \text{donde } x \in D \text{ y } y \geq 0, \end{aligned}$$

donde x e y son los vectores de decisiones con dimensiones $n\times1$ y $n\times1$, respectivamente y c, A, G, b son datos conocidos con dimensiones $n\times1$, $m\times n$, $m\times n$, $m\times1$, respectivamente. Las variables y son típicamente continuas y D abarca un dominio discreto.

Para cualquier asignación x^*, el problema auxiliar se formula como sigue:

$$\begin{aligned} &\text{SP: min} && cy \\ &\text{Sujeto a:} && Gy \geq b - Ax^* \qquad \text{donde } y \geq 0 \text{ y } Ax^* \text{ son constantes.} \end{aligned}$$

Una solución al dual de este problema lineal es una combinación lineal de sus desigualdades cuyo lado izquierdo es menor que cy (es decir u tal que $uG \leq c$) conduciendo así a una cota inferior: $cy \geq u(b - Ax^*)$. El teorema de dualidad indica que para una u^* minimizando $u(b - Ax^*)$, está cota es el óptimo de SP.

Se advertirá que la factibilidad dual es independiente de x^*, así que la desigualdad $cy \geq u^*(b - Ax)$ es válida para toda x. Sin embargo, cada vez que una nueva x_i^* es considerada, la correspondiente desigualdad puede ser agregada al problema maestro (MP) definido después de I iteraciones por:

$$\begin{aligned} &\text{MP: min} && z \\ &\text{Sujeto a:} && z \geq f(x)+u_i^*(b - Ax) \qquad \forall\ i < I \quad \text{donde } x \in D. \end{aligned}$$

Cada iteración i consiste de:

1. Encontrar una solución x_i^* de MP.
2. Encontrar una solución óptima y_i^* de SP y una solución óptima u_i^* de su dual.
3. Agregar el correspondiente corte de Benders a MP.

El algoritmo se detiene una vez que el problema maestro no pueda producir una solución mejoradora. Esto ocurrirá por lo menos una vez que toda x^* haya sido tratada. En otras palabras, cada iteración z_i^* es una cota inferior del problema original y (x_i^*, y_i^*) es una solución del problema con costo γ_i^*: el procedimiento se detiene una vez que $z_i^* = \min_i(\gamma_i^*)$.

Desde un punto de vista práctico, BD se puede aplicar eficientemente si el problema auxiliar (SP) es fácil de solucionar o estructurado especialmente y si los cortes conducen a una convergencia rápida del problema maestro. Sin embargo, es importante advertir que BD tiene la reputación de consumir mucho tiempo de cómputo, esto se debe en parte a la cantidad de cortes que son introducidos en el proceso antes de converger, y en parte debido a la dificultad del problema maestro.

Algunas aplicaciones importantes de BD las podemos ver en el trabajo de Florian [21] quien utilizó BD para programar el movimiento de máquinas ferroviarias y Richardson [60] quien aplicó el algoritmo para ruteo de aerolíneas. Fisher y Jaikumar [20] han discutido las ventajas de utilizar el algoritmo para problemas de ruteo de vehículos. Magnanti y Wong [48] propusieron una metodología para mejorar el rendimiento de BD cuando es aplicada a problemas enteros mixtos. En otro trabajo Magnanti [50] aplicó BD para resolver el problema de diseño de red no capacitada con aristas no dirigidas.

4.5 Descripción de la Metodología de Solución Propuesta

Una problemática palpable de nuestro modelo matemático es la función objetivo declarada en (4), que como se recordará es no-lineal, ahora bien, para atender esa dificultad de modelado nuestra metodología se apoya en una estrategia basada en escenarios, según la *RO*. El uso de escenarios facilita significativamente el manejo de nuestra función objetivo (4), ya que mediante esta estrategia podemos dividir a $DRCM_{UU}$ en problemas individuales, uno por cada escenario. La información generada por cada problema individual es conducida a optimizar la función objetivo del problema original (4).

La dificultad computacional de los problemas grandes de diseño de redes nos motivó a adoptar una versión relajada de la función objetivo (4), tal como hacen Kouvelis y

Yu en su problema de diseño de red con capacidades infinitas [43]. Por consiguiente, estamos interesados en encontrar *"diseños de red robustos"* los cuales estén dentro de un *P%* de la solución óptima para cualquier escenario de datos potencialmente realizable.

Formalicemos el concepto de *"diseño de red robusto"*. Si de la función objetivo denotada en (4) ignoramos el operador de maximización y fijamos un escenario dado, entonces la formulación resultante (la cual denotamos como $DRCM_S$) es el problema de diseño de red determinístico. La función objetivo a $DRCM_S$ es denotada por $Z_s(x,y)$ y la solución óptima por $(x^*(s),y^*(s))$. Denotemos a (x,y) como un diseño de red, donde $x = \left(x_{ij}^k\right)_{k\in K,(i,j)\in A}$, $y = \left(y_{ij}\right)_{(i,j)\in A}$, y permitamos que DR sea el conjunto de todos los diseños de red potenciales. Entonces, la solución óptima $(x^*(s),y^*(s))$ a $DRCM_S$ es tal que:

$$Z_s(x^*(s),x^*(s)) = \min_{(x,y)\in DR} Z_s(x,y) \ .$$

Para un conjunto de escenarios S, un diseño de red (x,y) es robusto si y solo si:

$$Z_s(x,y) - Z_s(x^*(s),y^*(s)) \le \frac{P}{100} Z_s(x^*(s),y^*(s)) \qquad \forall\, s \in S$$

donde *P/100* es un porcentaje de desviación con respecto a optimalidad.

Para resolver $DRCM_{UU}$ primeramente obtendremos una solución inicial que sea factible bajo cualquier escenario posible. La solución inicial es asignada a las variables binarias y_{ij}, lo cual hace que el problema resultante sólo en las variables de flujo x sea lineal. Este problema resultante puede ser descompuesto en s pequeños sub-problemas lineales de flujo multiproducto, uno por cada escenario. La solución óptima dual de cada sub-problema se usa para encontrar un nuevo conjunto de valores para las variables binarias. En lugar de generar cortes para problemas enteros como en BD, esas variables duales son combinadas para formar vecindades y se conduce una búsqueda en las vecindades generadas. El método de búsqueda está basado en una estrategia de memoria a corto plazo de TS y se basa en eliminación e inserción de aristas. Un nuevo conjunto de valores es seleccionado para las variables binarias y el procedimiento continúa hasta que algún criterio de terminación es alcanzado.

Por lo tanto, la metodología de solución diseñada para resolver $DRCM_{UU}$ es un algoritmo heurístico que utiliza estrategias de BD y TS y puede ser resumida en los siguientes pasos que serán explicados detalladamente en las siguientes secciones.

Paso 1: Obtención de una solución inicial.
Paso 2: Resolver los sub-problemas asociados a cada escenario.
Paso 3: Realizar búsqueda local mediante:
- 3.1 Eliminación de aristas.
  - 3.1.1 Cálculo del beneficio relativo de una arista.
  - 3.1.2 Eliminar aristas hasta provocar infactibilidad.
- 3.2 Inserción de aristas.
  - 3.2.1 Obtención de las q-rutas más cortas.
  - 3.2.2 Depuración.

Paso 4: Generación de diseños en Paso 3.1 y 3.2, evaluarlos.
Paso 5: Si el criterio de parada no ha sido satisfecho, regresar al Paso 2.

4.6 Paso 1: Obtención de una Solución Inicial Factible

El objetivo aquí es obtener un conjunto inicial de valores para las variables y_{ij} que constituya un diseño factible bajo cualquier escenario, esto es, que sea capaz de transportar exitosamente la demanda de cada producto sin importar el escenario que pueda presentarse.

El mecanismo para obtener esta solución inicial a $DRCM_{UU}$ es hacer una reducción al problema original. Un punto a nuestro favor es la suposición que se hace al usar escenarios para acaparar la aleatoriedad en el problema. La reducción consiste en generar un problema de diseño de red determinístico que esté ligado al problema original.

Si bien el problema de diseño de red multiproducto con capacidades en las aristas, aún en su versión determinista, sigue siendo un problema combinatorio de gran dificultad, podemos hacer uso de técnicas eficientes desarrolladas específicamente para este problema.

Como recordaremos la aleatoriedad que se presenta en $DRCM_{UU}$ se encuentra en los parámetros demanda ($d^k(s)$), y en los costos variables o de ruteo ($c_{ij}^k(s)$). Para garantizar que el diseño inicial que se obtenga sea factible se definirá un problema al que

denominaremos $DRCM_{factible}$, el cual tiene las siguientes características en sus parámetros de entrada:

- Los costos fijos (F_{ij}) asociados a cada arista son los mismos que en el problema original.
- Las capacidades (u_{ij}) asociadas a cada arista son las mismas que en el problema original.
- Las demandas que se toman son de acuerdo al siguiente criterio, con respecto al problema original $DRCM_{UU}$:

$$D^k = \max_{s \in S} \{d^k(s)\} \qquad \forall\, k \in K.$$

- Los costos variables ($c_{ij}^k(s)$) que se toman son de acuerdo al siguiente criterio, con respecto al problema original $DRCM_{UU}$:

$$C_{ij}^k = E(c_{ij}^k(s)), \qquad \text{donde } E(c_{ij}^k(s)) \text{ se calcula como:}$$

$$E(c_{ij}^k(s)) = \sum_{s \in S} P_s(c_{ij}^k(s)) \qquad \forall (i,j) \in A',\ k \in K.$$

El tomar como demanda del producto k la demanda mayor entre todos los escenarios, garantiza que si encontramos un diseño factible para este problema $DRCM_{factible}$, ese diseño es capaz de transportar las demandas de todos los productos en cualquier escenario de datos que se presente. Con respecto a los costos variables ($c_{ij}^k(s)$) se tomo el valor esperado de $c_{ij}^k(s)$ con el propósito de considerar un promedio ponderado de los costos variables incurridos a través de todos los escenarios.

El problema $DRCM_{factible}$ continúa siendo un problema entero mixto, pero de menor dimensión al original $DRCM_{UU}$, y como podemos observar presenta características completamente deteminísticas, esto es, sus parámetros obedecen a una sola entrada de datos. Para más detalle definimos $DRCM_{factible}$:

$DRCM_{factible}$

$$\min z = \sum_{k \in K} \sum_{(i,j) \in A'} C_{ij}^k x_{ij}^k + \sum_{\{i,j\} \in A} F_{ij} y_{ij} \tag{5}$$

sujeto a :

$$\sum_{\{j:(i,j) \in A'\}} x_{ij}^k - \sum_{\{j:(j,i) \in A'\}} x_{ji}^k = \begin{cases} D^k & \text{si } i = O(k) \\ -D^k & \text{si } i = D(k) \quad k \in K, i \in N. \\ 0 & \text{en otro caso.} \end{cases} \tag{6}$$

$$\sum_{k \in K} (x_{ij}^k + x_{ji}^k) \leq u_{ij} y_{ij} \qquad \{i,j\} \in A. \tag{7}$$

$$x_{ij}^k \geq 0 \qquad k \in K, (i,j) \in A'.$$

$$y_{ij} \in \{0,1\} \qquad \{i,j\} \in A. \tag{8}$$

Al resolver $DRCM_{factible}$ se obtiene una solución inicial a $DRCM_{UU}$ que es factible para cualquiera de los escenarios.

La solución a $DRCM_{factible}$, (x^0, y^0) donde $x^0 = \left(x_{ij}^k\right)_{k \in K, (i,j) \in A'}$, $y^0 = \left(y_{ij}\right)_{\{i,j\} \in A}$, será obtenida mediante un procedimiento especifico desarrollado a tal efecto. El procedimiento que se usa está basado en SS [14]. Por consiguiente, esta solución ya es de buena calidad para $DRCM_{factible}$.

4.7 Paso 2: Resolución de los Sub-problemas Asociados a Cada Escenario

Como ha sido mencionado, una vez que se dispone de un conjunto inicial de valores para las variables de diseño y_{ij}, el problema resultante (cuando se sustituyen estas variables en el problema original $DRCM_{UU}$) puede ser descompuesto en $|S|$ problemas de flujo multiproducto.

El propósito ahora es aprovechar la información que pueda ofrecer cada uno de los sub-problemas, quedando de esta forma s sub-problemas a estudiar. Para más detalle definimos el sub-problema de flujo a costo mínimo multiproducto (Minimum Cost Network Flow, MCNF, por sus siglas en inglés) que se origina en cada escenario al implementar el diseño $(y)^0$ obtenido al resolver $DRCM_{factible}$:

MCNF$_s$

$$\min\ z_s = \sum_{k\in K} \sum_{(i,j)\in A'} c_{ij}^k(s) x_{ij}^k(s) + \sum_{(i,j)\in A} F_{ij} y_{ij}^0 \tag{9}$$

sujeto a:

$$\sum_{\{j:(i,j)\in A'\}} x_{ij}^k(s) - \sum_{\{j:(j,i)\in A'\}} x_{ji}^k(s) = \begin{cases} d^k(s) & \text{si } i = O(k) \\ -d^k(s) & \text{si } i = D(k) \\ 0 & \text{en otro caso.} \end{cases} \quad k \in K, i \in N. \tag{10}$$

$$\sum_{k\in K} (x_{ij}^k(s) + x_{ji}^k(s)) \le u_{ij} y_{ij}^0 \qquad \{i,j\} \in A. \tag{11}$$

$$x_{ij}^k(s) \ge 0 \qquad k \in K, (i,j) \in A'. \tag{12}$$

Nótese que $\sum_{\{i,j\}\in A} F_{ij} y_{ij}^0$ es una constante y puede ser eliminado del análisis en cada escenario. Ahora bien, como se había aclarado, existirán tantos sub-problemas como escenarios, claro que al definir cada uno de los problemas MCNF$_s$ cada uno contendrá los parámetros correspondientes al escenario en estudio. Obsérvese que un conjunto inicial de valores ha sido fijado a las variables binarias y_{ij}, lo cual hace el sub-problema MCNF$_s$ lineal. La solución óptima dual de cada sub-problema, será usada para encontrar un nuevo conjunto de valores de las variables binarias y_{ij}. En lugar de generar cortes para un problema entero como se hace en BD, estas variables duales son combinadas para formar vecindarios de soluciones prometedoras y se conduce una búsqueda en los vecindarios generados. El método de búsqueda está basado en una estrategia de memoria a corto plazo de TS [26].

Para generar los vecindarios donde se conducirá la búsqueda se definirá lo que hemos llamado *"beneficio relativo de una arista"*. El cálculo del beneficio relativo de una arista está ligado con la estrategia de definir un orden adecuado para la eliminación e inserción de aristas. De dicho orden depende la búsqueda de soluciones robustas y de ello que se arrojen soluciones de buena calidad para DRCM$_{UU}$.

El cálculo del beneficio relativo se basa en los valores duales, marginales (π_{ij}) que se generan de resolver los s sub-problemas lineales MCNF$_s$ al fijar una solución $(y)^0$. Los valores duales, marginales (π_{ij}) en programación lineal (LP, linear programming, por sus siglas en inglés), representan una medida de aumento o disminución (según sea el caso) en

la función objetivo del primal cuando incrementamos una unidad en los parámetros del lado derecho asociados a cada una de las restricciones en el primal. En nuestro problema se utilizarán los valores duales (π_{ij}) asociados a la restricción de capacidad:

$$\sum_{k \in K} (x_{ij}^k(s) + x_{ji}^k(s)) \leq u_{ij} y_{ij} \qquad \{i, j\} \in A.$$

Estos valores duales indican el mejoramiento de la función objetivo logrado por cada unidad adicional de capacidad en cada arista. Debido a que el lado derecho de las restricciones de capacidad es multiplicado por una variable binaria, la capacidad no puede ser incrementada unitariamente. En otras palabras, la capacidad de una arista puede ser cero o u_{ij}. Sin embargo, nosotros usamos los valores duales como una valiosa aproximación para determinar el mérito de eliminar o agregar una arista $\{i,j\}$.

Los valores marginales (π_{ij}) asociados a la restricción de capacidad pueden ser negativos ó 0. Un valor negativo en las variables marginales (π_{ij}) nos indica que es una arista cuello de botella la que se pretende eliminar o insertar. Desde la perspectiva de conservación de factibilidad no es aconsejable eliminar alguna de las aristas que presenten estos valores, debido a que podría repercutir en infactibilidades. Sin embargo, si se trata de insertar aristas estas serían adecuadas ya que representarían un uso mayor de la capacidad por parte de los productos transportados.

Ahora bien, cuando un valor marginal (π_{ij}) es igual a 0, esto, nos indica que la arista que se pretende eliminar es holgada en capacidad y que podría no ocasionar infactibilidades en caso de decidir eliminarla. Para el caso de insertar aristas repercute en incrementar costos ya que no se aprovecha en su totalidad la capacidad de las aristas insertadas.

Debido a que hay bajo consideración un conjunto de escenarios, existe un precio sombra ($\pi_{ij}(s)$) para cada restricción de capacidad en cada uno de los s sub-problemas de programación lineal $MCNF_s$. En lugar de un solo precio sombra nosotros usamos un promedio ponderado de esos precios sombra resultantes de resolver los sub-problemas en cada escenario, donde los pesos son las probabilidades de ocurrencia de cada escenario potencialmente realizable. El cálculo resultante lo denominamos *"el precio sombra esperado"* y es expresado matemáticamente como sigue:

$$E(\pi_{ij}) = \sum_{s \in S} P_s(\pi_{ij}(s)) \qquad \forall \{i,j\} \in A.$$

Estos precios sombra pueden ser como indicamos, ya sea 0 (para exceso de capacidad) o negativos (cuello de botella).

Note que los precios sombra esperados se refieren solamente a los costos variables, con el objetivo de crear una medida que también tome en consideración el flujo (x_{ij}^k) y los costos fijos (F_{ij}) definimos el *beneficio relativo de una arista* (r_{ij}), como:

$$r_{ij} = \begin{cases} \dfrac{E(\pi_{ij})}{a_{ij}} & \text{si } E(\pi_{ij}) < 0 \\ a_{ij} & \text{si } E(\pi_{ij}) = 0 \end{cases} \tag{13}$$

donde el término a_{ij} se origina de la distribución de los flujos al asignar un diseño de red $(y)^0$ a $MCNF_s$ y se discute en la siguiente sección.

4.7.1 Término a_{ij}

El valor del término a_{ij} tiene mucha importancia para la obtención de buenos diseños robustos. El cálculo del término a_{ij} es información aprovechada de los s sub-problemas de distribución de flujo multiproducto ($MCNF_s$) y se origina de la distribución de flujos que se efectúa al asignar un diseño de red $(y)^0$ a $MCNF_s$.

La información es analizada desde la perspectiva de determinar cuánto es lo que pagamos en promedio por unidad de flujo transportado por una arista. De acuerdo a esta idea se define una razón que determine las aristas más baratas o caras dentro del diseño en estudio. Por lo tanto, se define al término a_{ij} como:

$$a_{ij} = \frac{F_{ij}}{\dfrac{\sum_{s \in S} \sum_{k \in K} x_{ij}^k(s) + x_{ji}^k(s)}{|S|}} \quad \forall \{i,j\} \in A. \tag{14}$$

Una vez obtenido el término a_{ij} (14), éste forma una parte importante dentro del cálculo del beneficio relativo de una arista (r_{ij}) ya que al realizar la operación en la ecuación (13), los resultados nos indicarán qué aristas son más propensas a provocar una

infactibilidad o adecuadas a insertar para recuperar factibilidad en alguno de los s sub-problemas $MCNF_s$.

4.8 Paso 3.1: Lista Candidata de Eliminación y Proceso de Eliminación de Aristas

Una vez obtenidos los valores de *beneficio relativo* (r_{ij}) para cada una de las aristas se puede definir un orden para este listado al cual se denomina como *Lista candidata de eliminación* (*LCE*). El orden de dicho listado será de mayor a menor, esto indicará el orden en que se deben eliminar las aristas del actual diseño de red $(y)^{actual}$, todo con la finalidad de hacer a nuestro diseño más barato y tratando de conservar la factibilidad por supuesto de los s sub-problemas. El sentido del orden de *LCE* de mayor a menor nos indica que las primeras aristas $\{i,j\}$ en el listado presentan valores marginales (π_{ij}) iguales a 0, por lo que entonces el beneficio relativo toma el valor a_{ij} que define qué aristas de las mencionadas presentan un costo fijo alto respecto a su flujo promedio bajo. Las aristas $\{i,j\}$ más propensas a provocar infactibilidades son conglomeradas al final del listado con la intención de no tomarlas en cuenta dentro del proceso de eliminación.

El *proceso de eliminación de aristas* se origina de la *Lista Candidata de Eliminación* (*LCE*) señalada anteriormente. Este proceso parte de *LCE* y se basa en dos etapas. La primera etapa consiste en realizar una exploración sobre las primeras ψ aristas de *LCE* (donde, ψ es un parámetro entero definido por el usuario). Si es posible eliminar una arista sin provocar infactibilidad en ninguno los s sub-problemas $MCNF_s$ entonces dicha arista es eliminada, de otro modo es mantenida en el diseño actual. Los experimentos computacionales han mostrado que considerar a ψ con un 60% de *LCE* brinda buenos resultados. En la segunda etapa, el remanente $|LCE|$-ψ de aristas es examinado para eliminación. Una vez que la etapa uno es efectuada estamos interesados en provocar infactibilidad en algunos de los $MCNF_s$, eliminando una o varias aristas de $|LCE|$-ψ.

Cada vez que una arista es eliminada (en el Paso 3.1) tenemos un nuevo diseño de red factible, del cual podemos indicar que es más barato que su antecesor. La eliminación de aristas de un diseño de red dado implica que al quitar una arista descartamos su costo

fijo (F_{ij}) y en esencia estos costos son los más relevantes dentro de los problemas de diseño de redes, ya que estos incluyen tarifas por infraestructura, ingeniería, construcción, etc. Dado que nuestro propósito radica en hacer diseños de red más baratos que satisfagan las restricciones de todos los escenarios potencialmente realizables, esta etapa forma una parte esencial dentro de la generación y exploración de soluciones robustas. Partir de una buena solución inicial $(x, y)^0$ y además hacerla más barata implica que los próximos cambios de aristas que entren dentro de nuestro diseño exploren otros entornos, y a su vez jueguen parte esencial de la búsqueda de nuevos diseños que sean más baratos pero sobre todo robustos.

4.8.1 Elementos de la Lista Tabú

Los elementos de esta lista lo conforman aquellas aristas que fueron eliminadas dentro del proceso de *eliminación de aristas* $\{i,j\}$ en el Paso 3.1. La finalidad es mantener un listado en memoria que nos presente qué aristas fueron eliminadas, y de este modo mantenerlas vetadas durante un cierto número de iteraciones antes de su posible reintegración en futuras exploraciones de otros entornos. La tenencia tabú (iteraciones de veto) es un parámetro de implementación.

4.9 Paso 3.2: Inserción de Aristas

El propósito de este paso es añadir aquellas aristas que puedan lograr que el diseño recupere la factibilidad en aquellos s sub-problemas donde la perdió, sin incrementar mucho el costo del diseño de la red.

Para cumplir con este objetivo definimos un diseño de red $(x, y)^{alt}$ al cual denominaremos como *alternativo*. Dicho diseño en un principio constará de todas las aristas potenciales del problema original, descartando aquellas que formen parte del actual diseño $(y)^{actual}$ y de la Lista Tabú (ξ) de la actual iteración.

La idea es decidir de entre todas las aristas que no forman parte del actual diseño, cuál o cuáles son las que van a entrar, esperando que se recupere la factibilidad de $MCNF_s$.

De este modo el *diseño alternativo* $(y)^{alt}$, se fijará en cada problema de distribución de flujo $MCNF_s$, y de allí es de donde aprovecharemos la información resultante para decidir qué aristas son las que serán insertadas para recuperar la factibilidad.

4.9.1 Detección de Infactibilidades

El objetivo de este punto es conocer cómo se provocó la infactibilidad. El daño provocado por las anteriores eliminaciones se refleja en cada sub-problema $MCNF_s$ al momento de intentar distribuir los productos por la red resultante. Por eso cuando decimos "incurrir en infactibilidad" significa que no se pudo satisfacer la demanda $(d^k(s))$ para alguno de los productos en alguno de los sub-problemas $MCNF_s$.

Para ello nuestra estrategia se basa en determinar en cuáles productos (k) de los s $MCNF_s$ se provocó infactibilidad, esto es, no se pudo transportar toda la demanda d^k hasta su destino. Con estos productos se formará un listado el cual se tendrá en cuenta para el proceso de inserción de aristas.

Como podemos apreciar es posible que el mismo producto (k) pueda ser mencionado en varias ocasiones pero para diferentes sub-problemas.

El análisis sobre las aristas candidatas se realiza a partir de la información arrojada por los s sub-problemas $MCNF_s$. Para ello se calculará el *beneficio relativo de una arista* (r_{ij}) de acuerdo a lo explicado y definido en las ecuaciones (13) y (14).

4.9.2 Obtención de las Q-rutas Más Cortas

Agregar aristas aisladas no necesariamente conduce a la recuperación de la factibilidad, por eso, una o más rutas serán insertadas en $(y)^{actual}$. Estas rutas son del origen al destino de cada producto cuya demanda no pudo ser transportada en $(y)^{acaual}$. Con el objetivo de hacer lo anterior, para cada uno de estos productos, las q-rutas más cortas serán obtenidas.

Cuando deseamos obtener una ruta más corta, primero es necesario establecer una medida de longitud de cada arista en la red. En nuestro caso, esta longitud debe mostrar el potencial adecuado de cada arista que permita a un diseño ser factible sobre todos los escenarios. La longitud de una arista considerará el *beneficio relativo de una arista* (r_{ij}) y la capacidad de la siguiente forma:

$$long_{ij} = \frac{r_{ij}}{u_{ij}} \qquad \forall \{i, j\} \in A.$$

Con estas longitudes serán determinadas las q-rutas más cortas para cada producto cuya demanda no pudo ser transportada en $(y)^{actual}$ según el listado de infactibilidades. Estas rutas son añadidas según se vayan necesitando con el objetivo de recuperar la factibilidad.

Las q-rutas más cortas se obtienen mediante un algoritmo específico desarrollado por Melián y Laguna [55], el cual es una implementación del algoritmo de Lawler [47]. Las aristas que conforman la primera ruta más corta son asignadas al actual diseño $(y)^{actual}$ y se verificará si se recupera la factibilidad a los s sub-problemas $MCNF_s$. Parte de la estrategia de obtener las q-rutas más cortas es mantener en memoria otras q-1 rutas más cortas, esto con el propósito de tener otras rutas para asignar en caso que la primera no logre recuperar la factibilidad. El conjunto de aristas correspondientes a las rutas que recuperan la factibilidad es almacenado en un listado denominado *a-agreg* el cual será utilizado dentro de una fase de depuración que se contempla para eliminar aristas no esenciales.

4.9.3 Depuración

Este proceso se establece como una estrategia para evitar iteraciones y disminuir el trabajo computacional al reiniciar el proceso de eliminación de aristas (Paso 3.1). Para ello aprovechamos la información arrojada por la comprobación de recuperación de la factibilidad al asignar rutas más cortas al sub-problema infactible $MCNF_s$ (Paso 3.2). De este modo analizamos la información arrojada por los sub-problemas y la aprovechamos para estudiar un poco más las aristas asignadas y enlistadas en *a-agreg*.

Nuestro propósito es determinar del listado *a-agreg* qué aristas son esenciales y cuáles no. El sentido de esencial lo definimos como la propiedad de una arista $\{i,j\}$ elemento de *a-agreg* para transportar flujo a lo largo de todos los s sub-problemas $MCNF_s$. De esta forma toda arista que transporte flujo a lo largo de todos los escenarios será etiquetada como esencial y por lo tanto quedará fija como parte del conjunto de aristas que logran recuperar la factibilidad. Aquellas aristas que no cumplan con esta condición se descartan y son eliminadas del diseño actual $(y)^{actual}$.

Dentro de los estudios que se realizaron se observó que realizar esta depuración reduce el número de iteraciones al retornar al Paso 3.1, por lo tanto, hay una disminución en el tiempo de cómputo y se observan mejores resultados respecto a no aplicar la depuración en la metodología. Para más detalles de este estudio ver el Apéndice D.

Para garantizar la recuperación de factibilidad a $MCNF_s$ se utiliza dicho diseño depurado, en caso de no recuperarse la factibilidad entonces nos avocamos a analizar las aristas que se descartaron de *a-agreg* y de estas se agregarán aquellas que pertenezcan a alguna de las trayectorias de los productos (k) que perdieron factibilidad. Como opción extraordinaria, si la inserción de estas aristas tampoco recupera la factibilidad a $MCNF_s$ se reasigna todo el listado *a-agreg* nuevamente.

4.10 Criterio de Parada

Uno de nuestros criterios de parada es cuando un diseño de red (y) se encuentre a un $P\%$ con respecto a la solución óptima de cada escenario (o una buena cota superior). Un segundo criterio de parada a considerar se basa en el número de iteraciones (oscilaciones) por parte de la metodología (el número de iteraciones es un parámetro de implementación).

CAPÍTULO 5
DISEÑO DEL EXPERIMENTO COMPUTACIONAL

5.1 Introducción

El objetivo principal de este capítulo es describir el diseño del experimento computacional que se realizó para validar la metodología propuesta así como del establecimiento de parámetros de la metodología para su eficiente ejecución.

Junto con lo anterior los principales aportes de este capítulo lo constituyen:

- La generación de un archivo manipulable que identifique a nuestros problemas bajo aleatoriedad.
- La creación de archivos con información del problema (útil para nuestros programas ejecutables).
- Implementación de archivos *.gms que son para uso de software de modelación y optimización que generará información de utilidad para varias de nuestras rutinas.

Cada una de las instancias que fueron creadas obedece a un entorno (o carpeta) especial que se detallará más adelante. También se mostrará la implementación del software de modelación GAMS 5.3 [38] para la generación de información a partir de los sub-problemas $MCNF_s$.

Se comenzará por detallar el establecimiento de parámetros de la metodología de búsqueda para continuar con un criterio de comparación, después con las características del generador aleatorio de instancias, parámetros del generador de instancias, características de instancias creadas, así como explicación de la *razón de capacidad* utilizada para clasificar

una instancia. Por último finalizaremos con la implementación de GAMS como herramienta de ayuda.

5.2 Parámetros del Algoritmo de Solución

En este punto se definen los valores de los parámetros utilizados por nuestra metodología de búsqueda para su efectiva ejecución. Para ello nos ayudamos de la evaluación del desempeño de los experimentos para determinar la influencia de unos o más parámetros en la calidad de la solución obtenida por la metodología.

Parámetro se define como cualquier variable controlable de un experimento que influye en el resultado final del mismo. Es bien sabido que el establecimiento correcto de los parámetros se reflejará en las soluciones de calidad al problema $DRCM_{UU}$.

A continuación mencionaremos y detallaremos los parámetros que intervienen en el algoritmo, y se definirá su valor o rango de valores para su efectiva ejecución.

a. Valor de ψ

b. Pertenencia a la Lista Tabú (ξ)

c. Oscilaciones (iteraciones de la metodología)

a. El valor de ψ está ligado a la exploración de la *LCE*, indicando cuántas aristas $\{i,j\}$ exploramos para determinar si son eliminadas ó no. Las pruebas computacionales revelaron que un 60% de *LCE* ofrece buenos resultados. Un porcentaje arriba de 75% reveló que analizar más aristas recae en efectuar trabajo de más. Ahora bien, un porcentaje por debajo de 50% de las aristas a explorar mostró que se dejan aristas potenciales a eliminar del diseño. Por lo tanto, valores adecuados a este parámetro los establecemos entre un 55%-65% para una correcta ejecución del algoritmo. En la Tabla 1 mostramos los resultados obtenidos al calibrar este parámetro para instancias de 30 nodos (con 30 y 50 productos, 10 y 20 escenarios), así como 50 nodos (con 30 y 50 productos, 10 y 20 escenarios). Hay que mencionar que fueron evaluadas 40 instancias de 30 nodos y 40 instancias de 50 nodos con sus respectivas características. Los porcentajes reportados indican la mayor desviación

porcentual de la implementación del diseño final en cada escenario donde el costo asociado se compara contra una cota superior del costo óptimo de cada escenario (el mecanismo para obtener la cota superior se describe en el punto 5.3 de este capítulo). Los porcentajes se determinan de la siguiente manera:

$$\%\ desviación\ porcentual = \max_{s \in S} \left\{ \frac{Z_s^R - Z_s^*}{Z_s^*} \times 100 \right\}$$

donde Z_s^R es el costo del diseño final evaluado en cada uno de los escenarios $s \in S$, Z_s^* representa el costo de la cota superior en el escenario $s \in S$. Para un problema en particular se evalúan todos los escenarios con la formula anterior y se obtiene la mayor desviación relativa para ese problema.

Nodos	Valores de ψ asignados						
	ψ=45%	ψ=50%	ψ=55%	ψ=60%	ψ=65%	ψ=70%	ψ=75%
30	2.45%	1.19%	-1.12	-1.70%	-1.70%	-1.70%	-1.70%
50	6.12%	5.74%	5.12%	3.63%	3.48%	3.48%	3.48%
General	4.29%	3.47%	2%	0.97%	0.89%	0.89%	0.89%

Tabla 1. Calibración de ψ

Los valores con signo negativo indican una mejora a favor del algoritmo desarrollado en este trabajo con respecto a una cota superior. El valor que se fijo a ψ fue de 60%. En la Tabla 2 se muestran los tiempos promedio, medidos como tiempo CPU en segundos, para cada uno de los valores ψ analizados.

Nodos	Tiempo (CPU en segundos)						
	ψ=45%	ψ=50%	ψ=55%	ψ=60%	ψ=65%	ψ=70%	ψ=75%
30	421	432	478	494	530	587	612
50	1134	1169	1209	1278	1298	1345	1354
General	778	801	844	886	914	966	983

Tabla 2. Tiempo CPU empleado por cada valor ψ evaluado

b. La pertenencia a la Lista Tabú (ξ) indica el número de iteraciones que las aristas que han sido eliminadas de un diseño quedan vetadas de volver a ser incluidas. Para

establecer el valor de este parámetro se realizó un estudio con valores de tenencia tabú de 2, 3, 4 y 5 iteraciones. Los resultados de esta experimentación se muestran en la Tabla 3. Se observó que en la mayoría de los casos, luego que una arista salía de la lista tabú era considerada nuevamente para integrarse al actual diseño $(y)^{actual}$. Se establece un valor de tenencia igual a 3 para este parámetro, ya que en la mayoría de las instancias analizadas fue donde se registro inserción de algún elemento de la lista tabú (ξ) y fue donde se presentaron los mejores resultados por parte de la metodología. En la Tabla 3 presentamos los resultados generados por este estudio.

	Tenencia Tabú (iteraciones de veto)			
Nodos	2	3	4	5
30	-0.23%	-1.17%	-0.95	-0.83%
50	4.53%	3.93%	4.74%	5.16%
General	2.15%	1.38%	1.90%	2.17%

Tabla 3. Calibración de la Tenencia Tabú

Al igual que en la Tabla 1, los porcentajes reportados indican la mayor desviación porcentual de la implementación del diseño final en cada escenario donde el costo asociado se compara contra una cota superior del costo óptimo de cada escenario. Además, los valores con signo negativo indican una mejora a favor del algoritmo desarrollado en este trabajo.

Los tiempos de cómputo para cada una de las iteraciones de veto evaluadas se muestran en la Tabla 4, se puede observar que los tiempos de cómputo no representan mucha diferencia al asignar un número de iteraciones en específico.

	Tiempo empleado para iteraciones de veto			
Nodos	2	3	4	5
30	549	543	556	568
50	1169	1198	1230	1156
General	859	870	893	862

Tabla 4. Tiempo CPU empleado para 2,3,4,5 iteraciones de veto

c. Para definir el parámetro referente al número de iteraciones del algoritmo tenemos que explicar el significado de oscilación dentro de nuestro estudio. Como recordaremos nuestra metodología utiliza dos pasos importantes de 4 contemplados en el algoritmo, un *Paso 3.1* que contempla eliminación de aristas y un *Paso 3.2* que se basa en inserción de aristas, los cuales se representan gráficamente en la Figura 1.

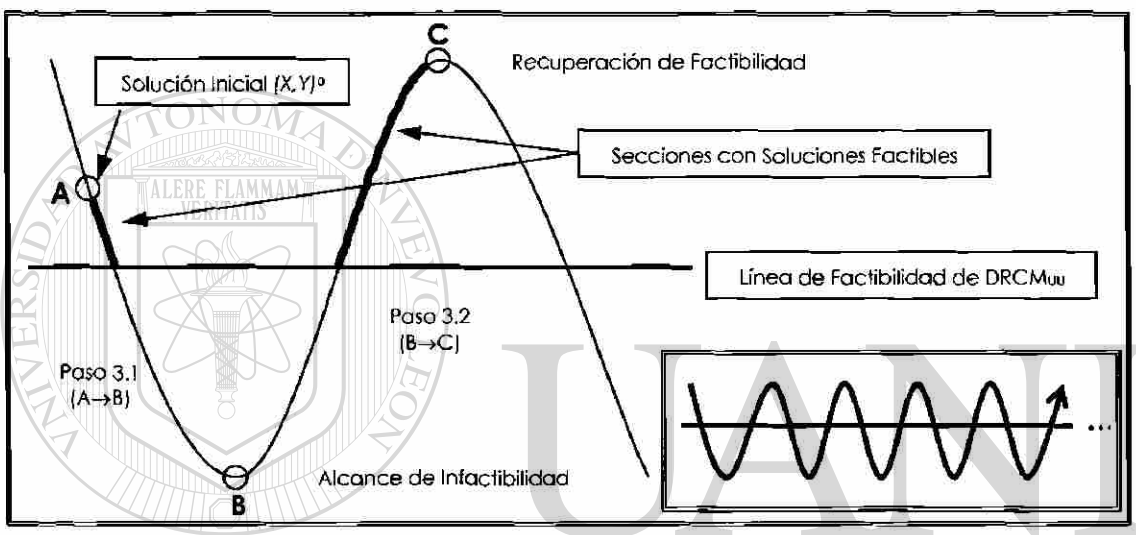


Figura 1. Representación de una oscilación (a→b→c)

De acuerdo a la Figura 1, una oscilación representa ejecutar una ocasión el *Paso 3.1* y el *Paso 3.2*, durante este recorrido el algoritmo genera soluciones factibles a lo largo de la sección de soluciones factibles. Por lo tanto, las oscilaciones representan el parámetro a definir en este punto. Como podemos observar, es necesario establecer un valor adecuado a este parámetro para no realizar oscilaciones de más que ocasionen un trabajo innecesario. Durante la experimentación se probaron valores de hasta 35 oscilaciones y en todos los casos las soluciones entregadas por el algoritmo no se mejoraban en comparación con menor número de oscilaciones. Ahora bien, se observó que valores entre 5 y 10 oscilaciones presentaron los mejores resultados por parte de la metodología. Dicho parámetro se establece en 10 oscilaciones.

5.3 Criterio de Comparación

La validación de los resultados de cualquier algoritmo heurístico tiene que estar sustentada por la comparación ya sea con trabajos anteriormente realizados, cotas superiores e inferiores, e incluso contra la solución óptima del problema. La comparación de los resultados contra el óptimo global del problema es en la mayoría de las veces una tarea difícil debido a que los problemas de optimización combinatoria en particular son de gran dificultad por su complejidad, ya que crece exponencialmente el tiempo de cómputo con el tamaño del problema. Recuérdese que en nuestro caso, la solución que el algoritmo entrega se pretende esté a un cierto *P%* de la solución óptima de cada escenario. Ahora bien, cuando comparamos contra una solución arrojada por un optimizador en realidad con lo que estamos comparando es la mejor solución entera encontrada después de cierto tiempo especificado al optimizador. Como estamos contemplando incertidumbre en el problema de $DRCM_{UU}$ y nuestra metodología se basa en escenarios, se vuelve una tarea complicada el comparar contra la mejor solución entera, ya que el tiempo de cómputo se vuelve una variable intratable al tratar de conseguir esos resultados por un optimizador.

Debido a esto nuestro criterio de comparación está definido por una cota superior de buena calidad que se obtiene mediante un procedimiento específico que fue desarrollado a tal efecto y está basado en la metaheurística Scatter Search [14]. El hecho de modelar la incertidumbre a través de escenarios facilita la implementación de SS en $DRCM_{UU}$, ya que cada escenario considerado en el problema original es visto como un problema de diseño de red determinístico al cual se obtiene una solución mediante SS. De este modo cuando evaluamos un problema $DRCM_{UU}$ se resolverán tantos problemas determinísticos como escenarios se consideren. De esa forma, cuando indicamos que la comparación es contra una cota superior nos estamos refiriendo a la solución de cada uno de los escenarios en el problema original $DRCM_{UU}$. Por lo tanto, se considera el siguiente criterio de evaluación para medir el desempeño del método que se implementará: comparación contra una cota superior. Cabe aclarar que se reportaron casos en donde la cota superior obtenida por el SS para un escenario dado a $DRCM_{UU}$, fue mejor que la mejor solución entera obtenida por el optimizador CPLEX después de 8 horas.

El mecanismo de comparación lo detallamos a continuación: El diseño final obtenido por nuestro algoritmo es implementado en cada escenario y el costo asociado lo comparamos con el costo de la cota superior del problema de diseño determinista asociado a dicho escenario. De esta forma se reporta la mayor desviación según el criterio de desviación robusta relativa. Estas desviaciones se determinan de la siguiente manera:

$$\%\ desviación\ porcentual = \max_{s \in S} \left\{ \frac{Z_s^R - Z_s^*}{Z_s^*} \times 100 \right\}$$

donde Z_s^R es el costo del diseño final evaluado en cada uno de los escenarios $s \in S$, Z_s^* representa el costo de la cota superior en el escenario $s \in S$. Para un problema en particular se evalúan todos los escenarios con la formula anterior y se obtiene la mayor desviación relativa para ese problema.

5.4 Generador Aleatorio de Instancias

Con la finalidad de probar nuestra metodología de búsqueda es necesario contar con una base de instancias de prueba. Para generar estas instancias se diseñó e implementó un procedimiento que genera instancias (y a la vez entornos o carpetas) de red en forma aleatoria. Como se ha explicado anteriormente, hasta donde conocemos, el problema $DRCM_{UU}$ no ha sido abordado en la literatura, esto implica que no contamos con una base de instancias de red que sea de ayuda para abordar nuestro estudio, incluso sobre el cual se pudiese realizar un estudio comparativo entre metodologías empleadas. Por ello es de nuestro interés crear una base de instancias de red bajo un formato adecuado que permita un fácil uso para investigaciones futuras de este problema. La generación de instancias así como la razón de capacidad de cada arista y densidad de la red constituyen una adaptación del trabajo desarrollado por De-Alba [14].

El generador crea un archivo en texto (ASCII) con diferentes registros. Por lo tanto cada instancia del problema $DRCM_{UU}$ poseerá un archivo único que indique la

información particular y las características propias de la instancia. El nombre de dicho archivo queda abierto al usuario, pero es recomendable usar una nomenclatura que contenga información sobre las características de la red que representa, por ejemplo el número de nodos, productos, escenarios, densidad de red, etc.

El archivo generado será usado como entrada para los diversos programas que se han elaborado, particularmente para el programa de obtención de una solución inicial. Es importante señalar el carácter general de dicho archivo, pues su formato puede ser utilizado como base en el desarrollo e implementación de otros procedimientos e investigaciones futuras.

Una de las características del generador de instancias es generar redes con diferentes grados de capacidad y densidad. Esto se hizo con la finalidad de obtener de una instancia de red instancias adicionales con el mismo número de nodos, productos, escenarios, costos fijos pero con diferentes características de capacidad o bien densidad de red.

5.4.1 Formato de Instancia de Red $DRCM_{UU}$

El formato del archivo que contiene las instancias de $DRCM_{UU}$ posee 4 diferentes tipos de registro.

Registro tipo 1. Está compuesto por las características generales de la red.

Registro tipo 2. Características generales de cada arista.

Registro tipo 3. Costos variables por producto y escenario para cada arista.

Registro tipo 4. Información de los productos (demanda, origen y destino) para cada escenario.

El archivo es generado apegándose al modelo siguiente:

Registro tipo 1: *printf*("%d %d %d %d \n", *N, M, K, S*);

printf("%f \n", *prob(s)*);

Ahora bien para cada arista {i, j} de la red tenemos:

Registro tipo 2: *printf*("%d %d %d %d \n", *i, j, CF, Cap*);

Para cada producto *k* en la arista {i, j} y escenario s tenemos:

Registro tipo 3: *printf*("%d %d %d \n", *k, s, CV*);

Por último para cada producto *k* en la red:

Registro tipo 4: *printf*("%d %d %d \n", *k, Or, De*);

printf("%d %d \n", *s, Dem*);

Donde:

N es entero e indica cantidad de nodos.

M es entero e indica cantidad de aristas.

K es entero e indica cantidad de productos.

S es entero e indica número de escenarios.

Prob es flotante e indica la probabilidad de ocurrencia de cada escenario.

i es entero e indica nodo origen de la arista.

j es entero e indica nodo destino de la arista.

CF es entero e indica costo fijo por utilización de la arista.

Cap es entero e indica capacidad de la arista.

k es entero e indica producto.

s es entero e indica escenario.

CV es entero e indica costo variable de la arista para el producto k en el escenario s.

Or es entero e indica origen del producto k.

De es entero e indica destino del producto k.

Dem es entero e indica demanda del producto k en el escenario s.

Nótese que en un archivo de red, existirá un sólo registro de tipo 1; habrá tantos registros tipo 2 como aristas posea la red; del registro tipo 3, por cada arista existirán tantos registros como productos circulen en la red y escenarios se contemplen, por lo que habrá un total de $K * M * S$ registros del tipo 3; finalmente, del registro tipo 4 existirán tantos registros como productos y escenarios tenga la instancia, por lo que habrá un total de $K * S$ de registros del tipo 4. Para más detalle observar en el Apéndice A un ejemplo de instancia.

5.4.2 Parámetros del Generador de Instancias

A continuación presentamos los parámetros del generador de instancias:

N — Entero que representa la cantidad de nodos en la red. Este valor es entrado directamente al generador según la cantidad de nodos que se desee posea la red.

K — Entero que representa la cantidad de productos que transportarán su demanda por la red. Este número es entrado directamente al generador según el número de productos que se desee transportar en la red.

S — Entero que representa la cantidad de escenarios a considerar. Este número es entrado directamente al generador según el número de escenarios que se desee contemplar para el problema en estudio.

D — Entero que representa la demanda máxima. Sin afectar la generalidad del estudio, se ha predefinido este valor entre 50 y 100.

CV — Entero que representa el costo variable. Predefinido entre 50 y 100.

P — Número fraccionario que representa la probabilidad de selección de una determinada arista en la red. Este valor se acerca a cero para obtener redes dispersas, o bien, a uno, para obtener redes densas. Se predefinió a 0.75 y 0.50 para considerar redes densas y medianamente densas.

LI — Número fraccionario que representa el límite inferior del rango de capacidad. Es una fracción con relación a la constante A detallada más abajo. Este valor se usa conjuntamente a LS explicado más abajo. Se predefinió a 0.8

LS — Número fraccionario que representa el límite superior del rango de capacidad. Es una fracción con relación a la constante A detallada más abajo. Este valor se usa conjuntamente a LI. Predefinido a 1.2.

A Número entero que representa la constante de capacidad. Este valor puede incrementarse a fin de obtener redes holgadas en capacidad. Predefinido a 1.

5.4.3 Instancias Creadas

Las instancias fueron generadas aleatoriamente de acuerdo a la densidad de red que se especifique, así como a qué tan ajustadas se deseen las capacidades de sus aristas. En la Tabla 5 reflejamos las características de importancia.

Características	
% Densidad	Capacidad
50%	Holgada (300-360 unidades)
50%	Restringida (140-180 unidades)
75%	Holgada (200-260 unidades)
75%	Restringida (90-130 unidades)

Tabla 5. Características de importancia en las instancias

Para cada una de las parejas de características declaradas se generaron un total de 5 instancias, variando además la cantidad de nodos (30 y 50), de productos (30 y 50) y de escenarios (10 y 20).

En la Tabla 6 se muestran las instancias que se generaron al igual que sus características generales. En la primera columna indicamos el número de nodos con que cuenta, en la segunda columna el número de aristas, en la tercera columna el número de productos, en la cuarta columna el número de escenarios a considerar en la instancia en estudio, en la quinta columna indicamos cómo se comporta la capacidad de las aristas (una H indica que el problema es holgado en capacidad y una R indica que es restringido en capacidad) y por último la sexta columna nos señala la cantidad de instancias generadas para cada una de las características generales.

Como podemos observar en la Tabla 6 se generaron un total de 160 instancias para realizar nuestro estudio.

Nodos	Aristas	Productos	Escenarios	Capacidad	Instancias
30	217	30	10	H	5
				R	5
			20	H	5
				R	5
		50	10	H	5
				R	5
			20	H	5
				R	5
	327	30	10	H	5
				R	5
			20	H	5
				R	5
		50	10	H	5
				R	5
			20	H	5
				R	5
50	613	30	10	H	5
				R	5
			20	H	5
				R	5
		50	10	H	5
				R	5
			20	H	5
				R	5
	919	30	10	H	5
				R	5
			20	H	5
				R	5
		50	10	H	5
				R	5
			20	H	5
				R	5
				TOTAL	160

Tabla 6. Instancias generadas

Ahora bien, al ejecutar el programa generador de instancias bajo las características antes descritas, lo que físicamente se está generando son entornos o carpetas con las instancias agrupadas según tipo de capacidad (holgado o restringido). Como se mencionó anteriormente, los parámetros introducidos directamente al programa son nodos, productos y escenarios, además que internamente podemos manipular la densidad de red así como el

número de instancias por crear, por ello cuando evocamos este programa parte de las características ya las estamos definiendo.

De modo que, una vez que evoquemos el programa generador de instancias tendremos una carpeta por cada instancia que analicemos, esto, ya que aparte de la generación del archivo de texto (ASCII) de la instancia de red, también creamos archivos que son necesarios para la ejecución del algoritmo. Tales archivos contienen información sobre costos fijos por cada arista definida en una instancia, probabilidades de ocurrencia de los escenarios, archivos de cada una de las instancias en formato GAMS 5.3 [38], capacidades de las aristas, todo esto información útil para la ejecución correcta de cada una de las rutinas del algoritmo.

La perfecta diferenciación entre cada una de las instancias creadas hace necesario acomodar cada instancia en una carpeta individualmente, ya que el aspecto de incertidumbre genera una inmensidad de información que al imaginar mezclarla con otra instancia en un mismo entorno o carpeta causaría una confusión en el manejo de información.

5.4.4 Razón de la Capacidad de una Arista y Densidad de la Red

Como se sabe el número máximo de aristas que puede existir en la red es N(N-1)/2, esto si la red fuera completa, es decir con su máxima densidad posible. Para cada arista que puede existir en la red, se obtiene un número aleatorio entre 0 y 1. Si este número es mayor o igual a P, la probabilidad de selección, dicha arista es incluida como arista potencial en la red, de otra forma, es desechada. Ahora bien, en el generador de instancias indicamos la densidad de red sobre la cual vamos a realizar el estudio, como se aclaró antes esta se fijó en 50% y 75%.

Para determinar el grado en capacidad, tanto de redes holgadas como restringidas se aplicó la siguiente razón de capacidad [14]:

$$\Im = \frac{\sum_k \sum_s d^k(s) MF^k(s)}{\left(\sum_k \sum_s d^k(s)\right)^2},$$

donde $MF^k(s)$ es el máximo flujo entre nodo origen y destino asociado al producto k en el escenario s. Cuando $\Im$ toma valores menores a 1, la instancia se considera restringida en capacidad, y cuando $\Im$ se incremente más allá de 1 la red se hace más holgada. Las instancias fueron ajustadas para asegurar la factibilidad del problema. La idea tras esta razón es el hecho que una red factible debe satisfacer la siguiente relación:

$$MF^k(s) \geq d^k(s) \; \forall \; k \in K, \; s \in S.$$

Entonces para una red factible $(\Sigma_{k\in K}\Sigma_{s\in S} MF^k(s)) \; / \; (\Sigma_{k\in K}\Sigma_{s\in S} d^k(s)) \geq 1$. Sin embargo, debido a que estamos interesados en medir la restricción en capacidad de una red factible, esta razón fue escalada para que estuviera cercana a 1 para redes restringidas, entonces para cada producto k y escenario s, $MF^k(s)$ se afectó por la importancia relativa de la demanda de este producto en un escenario respecto a la demanda total de los productos en todos los escenarios, esto es, $d^k(s)/(\Sigma_{k\in K}\Sigma_{s\in S} d^k(s))$. A medida que esta razón se aleja de 1, la red se considera menos restringida en capacidad.

5.4.5 Implementación de MCNF$_s$ en GAMS/CPLEX

GAMS [38] es un modelador algebraico del cual nos auxiliamos para hacer la representación matemática de los sub-problemas MCNF$_s$ (problemas lineales). Este software modela los s sub-problemas lineales y se especifica que sean resueltos con el optimizador CPLEX. CPLEX [8] es un poderoso optimizador para resolver problemas lineales y enteros de gran escala de manera eficiente y rápida. El programa GAMS/CPLEX cuenta con diversas opciones algorítmicas para resolver los problemas lineales, tales como el método simplex primal, simplex dual, método de puntos interiores. En nuestro caso seleccionamos utilizar el *Algoritmo Simplex Primal* para resolver los problemas lineales MCNF$_s$.

El uso de GAMS como herramienta modeladora de este problema es debido a su ambiente hasta cierto grado sencillo de realizar estas tareas complejas. Además, la

necesidad de generar información para nuestras rutinas en C motivó su elección como software de modelación.

Para ejecutar cada sub-problema $MCNF_s$ en una instancia dada, en GAMS se hace necesario guardar el archivo con extensión `*.gms` bajo un formato especifico pero sencillo. Una de las aplicaciones de interés que se utilizan de GAMS es que si un escenario de los sub-problemas $MCNF_s$ es infactible, entonces abortamos ese escenario no permitiendo más iteraciones. La generación de información para nuestras rutinas a partir de los sub-problemas $MCNF_s$ son los costos marginales ($\pi_{ij}(s)$), flujos ($x_{ij}^k(s)$), y valores objetivos (Z_s) . Se utiliza un solo archivo `*.gms` para declarar un sub-problema $MCNF_s$ con s escenarios posibles, mediante una estructura definida. Cada una de estas aportaciones por parte del modelador GAMS está detallada en el Apéndice B.

Como se mencionó, numerosas opciones del optimizador CPLEX pueden ser manipuladas por parte del usuario, una de ellas está relacionada con el uso de información de anteriores ejecuciones para utilizar en posteriores corridas y no comenzar a iterar de cero. En nuestro problema, cuando comenzamos una corrida digamos del escenario 1 bajo un diseño dado, para el escenario 2 y los siguientes escenarios no es necesario empezar a correr de inicio debido a que se aprovechan bases para encontrar la solución óptima de los siguientes escenarios. La opción dentro de CPLEX que nos facilita esta tarea es `bratio=1`, que es detallada al igual que otras opciones en el Apéndice B.

5.4.6 Implementación Computacional

Para la programación de todas las rutinas necesarias se utilizó lenguaje C con el compilador de Sun cc versión Forte y fueron ejecutadas en una estación de trabajo SUN™ Ultra 10, con sistema operativo Solaris™ versión 7, que permitió la realización de las pruebas. Cada una de las rutinas empleadas y el mecanismo utilizado para ejecutar la metodología de búsqueda se detalla en el Apéndice C. Las rutinas empleadas en este trabajo de tesis se pueden accesar desde la siguiente dirección URL:

http://yalma.fime.uanl.mx/~pisis/

CAPÍTULO 6
RESULTADOS COMPUTACIONALES

6.1 Introducción

En este capítulo se presenta la evaluación computacional realizada para validar la efectividad de la metodología presentada en el Capítulo 4. El número y características de las instancias, así como el proceso de generación fueron expuestos en el capítulo anterior. El criterio de evaluación considerado, como antes se mencionó, consiste en implementar el diseño final obtenido por nuestro algoritmo en cada escenario y el costo asociado lo comparamos con el costo de la cota superior óptima de los escenarios del problema. De esta forma se reporta la mayor desviación relativa. La solución factible obtenida para cada escenario es de buena calidad y fue obtenida mediante un procedimiento basado en SS [14].

6.2 Evaluación del Desempeño del Algoritmo de Solución

En este punto nos enfocamos a la evaluación de la eficiencia de la metodología de búsqueda propuesta. La evaluación de resultados se presenta con diferentes agrupaciones de las instancias generadas, considerando agrupaciones según cantidad de nodos, nivel de capacidad, cantidad de productos, de escenarios y densidad. Por otra parte, se consideran agrupaciones por nodos-densidad, nodos-escenarios, nodos-productos, nodos-capacidad, densidad-escenarios, densidad-productos, densidad-capacidad, escenarios-productos, escenarios-capacidad y productos-capacidad. Esto se hace así con el objetivo de distinguir qué características influyen más fuertemente en el desempeño del algoritmo.

Las tablas que muestran los resultados indican los siguientes rubros: *Red*, según el tipo de agrupación; *Final*, indica el mayor porcentaje de desviación relativa del costo del diseño final implementado en cada escenario respecto al costo de la cota superior óptima de los escenarios del problema original. Valores con signo negativo reportados en este rubro

indican una mejora a favor de la metodología con respecto al criterio de comparación. Por último el rubro *Tiempo*, nos indica el tiempo CPU en segundos que la metodología de búsqueda empleó para obtener la solución robusta. Hay que aclarar que los tiempos reportados incluyen la obtención de la cota superior. Los tiempos de ejecución fueron obtenidos con el comando `time`. En cada caso, los porcentajes fueron hallados promediando sobre todas las instancias que posean las características especificadas.

Red	Final	Tiempo
30 nodos	-2.1%	552
50 nodos	3.6%	1176
Total	0.8%	864

Tabla 7. Comparación cuando las instancias son agrupadas por nodos

En la Tabla 7 se muestran las instancias agrupadas por el número de nodos. Las instancias que mejor desempeño obtuvieron son las redes de 30 nodos respecto a las de 50 nodos. Con esto podemos indicar que el número de nodos afecta el desempeño de la metodología propuesta. Podemos observar que el tiempo de CPU empleado por la metodología en las instancias de 50 nodos se incrementa casi al doble respecto a las instancias de 30 nodos.

Red	Final	Tiempo
Holgada	-0.1%	831
Restringida	1.7%	898
Total	0.8%	864

Tabla 8. Comparación cuando las instancias son agrupadas por capacidad

En la Tabla 8 se muestran las instancias agrupadas según el tipo de capacidad de la red: holgada o restringida. Los resultados obtenidos en las instancias holgadas son mejores respecto a aquellas instancias con capacidad restringida. Esto debido a que la capacidad de una arista en una instancia holgada es mayor. Esta característica de poseer mayor capacidad en una arista ayuda al momento de optimizar el problema, por ejemplo, es posible distribuir el flujo de más productos por una arista respecto a una instancia

restringida (ya que si fuera restringida, quizás se necesitaría pagar el costo fijo de una arista más para distribuir el flujo), y esto en parte hace disminuir los costos variables, que aunque no son los más significativos sí contribuyen al costo total del problema.

Red	Final	Tiempo
30 productos	3.1%	487
50 productos	-1.5%	1241
Total	0.8%	864

Tabla 9. Comparación cuando las instancias son agrupadas por productos

En la Tabla 9 se muestran las instancias agrupadas según el número de productos. A diferencia de los resultados anteriores podemos observar en la columna de *Final* que las instancias con más productos (50 productos) presentan mejores resultados que aquellas con pocos productos (30 productos). Sin embargo podemos ver que aunque las instancias de 50 productos presentan mejores resultados, sus tiempos de CPU se ven incrementados casi 3 veces respecto a las instancias de 30 productos, esto indica la dificultad de resolver instancias bajo mayor cantidad de productos. Los mejores resultados de las instancias de 50 productos puede ser atribuida a que en un problema dado son aprovechadas de mejor manera las aristas elegidas para pertenecer en la red respecto a uno con 30 productos y de esta forma son abaratados los costos.

Red	Final	Tiempo
10 escenarios	0.8%	585
20 escenarios	0.7%	1143
Total	0.8%	864

Tabla 10. Comparación cuando las instancias son agrupadas por escenarios

En la Tabla 10 se muestran las instancias agrupadas de acuerdo a la cantidad de escenarios. Se puede observar que no existe mucha diferencia entre los resultados de una instancia de 10 escenarios y una de 20 escenarios, pudiéndose concluir que la cantidad de escenarios no afecta significativamente el desempeño de la metodología en cuanto a la calidad de las soluciones obtenidas. Un aspecto notable se refleja en la columna *Tiempo*: las

instancias de 20 escenarios emplearon casi el doble de tiempo a las instancias de 10 escenarios. Como era de esperarse, se reafirma que a instancias con mayor número de escenarios el tiempo de cómputo se incrementa.

Red	Final	Tiempo
50%	1.0%	857
75%	0.5%	871
Total	0.8%	864

Tabla 11. Comparación cuando las instancias son agrupadas por densidad

En la Tabla 11 se puede apreciar el agrupamiento de las instancias según la densidad de red. Este rubro como podemos observar no presenta mucha diferencia ni en cantidad de resultados ni en tiempo de cómputo entre instancias con 50% y 75% de densidad. Con esto podemos deducir que esta característica en las instancias no influye significativamente en el desempeño de la metodología, aunque puede apreciarse que mientras más densa es la red se logran mejores resultados puesto que hay más opciones a elegir.

Red	Final	Tiempo
30n-30k	1.9%	319
30n-50k	-6.1%	785
50n-30k	4.3%	655
50n-50k	3.0%	1697
Total	0.8%	864

Tabla 12. Comparación cuando las instancias son agrupadas por nodos – productos

A partir de la Tabla 12 mostramos el agrupamiento de resultados por parejas de característica. En esta tabla se muestra una comparación cuando las instancias son agrupadas por nodos-productos. Podemos apreciar que cuando fijamos los nodos a 30 y variamos la cantidad de productos los resultados son mejores a mayor cantidad de productos, este mismo comportamiento lo podemos observar cuando fijamos los nodos a 50 y hacemos variar los productos. Sin embargo, el tiempo de cómputo en instancias de 30n-50k se ve incrementado casi al doble respecto a las instancias 30n-30k. El

comportamiento del tiempo en las instancias 50n-50k es similar, se incrementa casi al triple respecto a las instancias 50n-30k. Fijando la cantidad de nodos, los buenos resultados en las instancias de 50 productos pueden ser atribuidos a que en un problema dado son aprovechadas de mejor manera las aristas elegidas para pertenecer en la red respecto a uno con 30 productos y de esta forma son abaratados los costos. Cuando fijamos la cantidad de productos y variamos los nodos los resultados son mejores a menor cantidad de nodos, concluyendo así que la cantidad de nodos afecta el desempeño de la metodología de búsqueda. El tiempo de cómputo al fijar la cantidad de nodos y variar los productos se ve incrementado al aumentar la cantidad de productos, es notable ver que el algoritmo requiere un esfuerzo computacional mayor para distribuir una mayor cantidad de productos.

Red	Final	Tiempo
30n-10s	-2.0%	371
30n-20s	-2.2%	733
50n-10s	3.7%	800
50n-20s	3.6%	1553
Total	0.8%	864

Tabla 13. Comparación cuando las instancias son agrupadas por nodos – escenarios

En la Tabla 13 se presenta la agrupación de las instancias respecto a la cantidad de nodos y número de escenarios. Notemos que las soluciones obtenidas por la metodología de búsqueda pierden calidad cuando se incrementa el número de nodos, concluyendo nuevamente en esta comparación que el número de nodos afecta de desempeño del algoritmo. Respecto al tiempo de cómputo empleado se puede observar que son necesarios más recursos para instancias de tamaño mayor.

Red	Final	Tiempo
30n-50%	-1.4%	520
30n-75%	-2.8%	584
50n-50%	3.4%	1195
50n-75%	3.9%	1157
Total	0.8%	864

Tabla 14. Comparación cuando las instancias son agrupadas por nodos – densidad

En la Tabla 14 se muestra el agrupamiento de las instancias por nodo-densidad. Los mejores resultados son presentados por las instancias de 30n-75% respecto a las instancias de 30n-50%, el tiempo de cómputo entre estas instancias es similar. En las instancias mayores, 50n-50% y 50n-75%, se puede observar en la columna *Final* que presentan se presenta una disminución en la calidad de los resultados obtenidos y en cuanto al tiempo de cómputo este se ve incrementado al doble respecto a las instancias de 30 nodos. Nuevamente se observó que a medida que aumenta la cantidad de nodos, también crece la diferencia de resultados.

Red	Final	Tiempo
30n-Holg.	-3.9%	536
30n-Rest.	-0.3%	568
50n-Holg.	3.6%	1126
50n-Rest.	3.6%	1227
Total	0.8%	864

Tabla 15. Comparación cuando las instancias son agrupadas por nodos – capacidad

En la Tabla 15 se presentan las instancias agrupadas por número de nodos y capacidad. El comportamiento es similar a tablas anteriores cuando se incrementa la cantidad de nodos. Con respecto a las capacidades, se observó que a medida que una red es más restringida también crece la diferencia de resultados, por lo tanto, se presentan mejores resultados en instancias holgadas. Esta mejoría de resultados en una instancia holgada respecto a una restringida se debe a que en una instancia holgada la capacidad de una arista puede ser aprovechada de mejor manera para distribuir los productos que son transportados a través del diseño final

Red	Final	Tiempo
50%-10s	1.0%	582
50%-20s	1.0%	1132
75%-10s	0.7%	588
75%-20s	0.4%	1154
Total	0.8%	864

Tabla 16. Comparación cuando las instancias son agrupadas por densidad - escenarios

En la Tabla 16 las instancias son agrupadas por densidad-escenarios. Se presentan resultados similares entre las instancias de 50%-10s y 50%-20s, el tiempo de cómputo entre estas se ve incrementado al doble en las instancias con 20 escenarios. Respecto a las instancias con un 75% de densidad, los problemas con 75%-20s presentaron ligeramente mejores resultados respecto a los problemas con 75%-10s. En cuanto al tiempo de cómputo el comportamiento permaneció igual al caso de 50% de densidad; las instancias con 20 escenarios incrementaron al doble su tiempo de cómputo respecto a las instancias de 10 escenarios. Esto reafirma la conclusión a la que llegamos al observar la Tabla 10, el número de escenarios no influye en la calidad de las soluciones sino en el tiempo de cómputo requerido para alcanzarlas.

Red	Final	Tiempo
50%-30k	3.4%	517
50%-50k	-1.4%	1198
75%-30k	2.7%	458
75%-50k	-1.7%	1284
Total	0.8%	864

Tabla 17. Comparación cuando las instancias son agrupadas por densidad - productos

En la Tabla 17 mostramos las instancias agrupadas por densidad-productos. Los mejores resultados son observados en las instancias con mayor densidad y más número de productos (75%-50k), aunque el tiempo de cómputo es mayor respecto a las demás instancias. Los mejores resultados presentados en estas instancias respecto a las demás es debido a que hay una mayor densidad de red, hay mejores opciones de elección de aristas respecto a otras instancias con menor densidad (ya que cuentan con ciertas aristas clave para el problema). En cuanto a los productos es notable ver que al momento de optimizar, las aristas que son elegidas para estar en la red son aprovechadas mejor que en una instancia con menor cantidad de productos, esto por lo tanto reduce significativamente el costo total del problema.

Red	Final	Tiempo
50%-Holg.	-0.015%	800
50%-Rest.	2.0%	914
75%-Holg.	-0.2%	861
75%-Rest.	1.3%	881
Total	0.8%	864

Tabla 18. Comparación cuando las instancias son agrupadas por densidad - capacidad

En la Tabla 18 agrupamos las instancias respecto la densidad-capacidad. Podemos ver que el comportamiento de la densidad es el mismo obtenido en tablas anteriores, a mayor densidad se presentan mejores resultados, esto debido a que hay más elección de aristas a pertenecer a la red. Se reafirma el comportamiento de instancias holgadas al presentar mejores resultados respecto a aquellas que son restringidas. Los tiempos de cómputo podemos apreciar que son similares en las 4 agrupaciones.

Red	Final	Tiempo
10s-30k	2.9%	331
10s-50k	-1.2%	840
20s-30k	3.2%	643
20s-50k	-1.9%	1643
Total	0.8%	864

Tabla 19. Comparación cuando las instancias son agrupadas por escenarios – productos

En la Tabla 19 se presentan las instancias agrupadas por escenarios-productos. Las instancias con mayor cantidad de escenarios (20) y mayor cantidad de productos (50) fueron la que presentaron los mejores resultados respecto a las tres agrupaciones restantes, aunque se puede observar que su tiempo de cómputo es mayor a las demás agrupaciones. Se puede observar que las instancias con 50 productos y variar los escenarios presentaron los mejores resultados respecto a las instancias con 30 productos. Esto sin duda reafirma que a mayor cantidad de productos es posible distribuirlos mejor en un diseño de red respecto a pocos productos, ya que el costo fijo en un arco puede ser compartido entre más productos. Podemos observar que el tiempo de cómputo se incrementa a mayor cantidad de productos.

Red	Final	Tiempo
10s-Holg.	0.1%	555
10s-Rest.	1.6%	615
20s-Holg.	-0.3%	1106
20s-Rest.	1.7%	1180
Total	0.8%	864

Tabla 20. Comparación cuando las instancias son agrupadas por escenarios - capacidad

En la Tabla 20 se puede apreciar el agrupamiento de las instancias por escenarios-capacidad. Se reafirma lo de tablas anteriores, los escenarios no influyen en el comportamiento de la metodología, pero al variar la capacidad es notable ver que se presentan mejores resultados en aquellas instancias que son holgadas. Los mejores resultados son reportados en las instancias 10s-Holg. y 20s-Holg.. En problemas con capacidad holgada se observa que se obtienen mejores resultados respecto a instancias con capacidad restringida, debido a tener una capacidad amplia en las aristas y utilizarla para distribuir el flujo de varios productos, en vez de emplear otra arista e incurrir en pago de costos fijos como en los problemas restringidos. Vale la pena aclarar que los costos fijos son los más relevantes (caros).

Red	Final	Tiempo
30k-Holg.	3.0%	457
30k-Rest.	3.1%	517
50k-Holg.	-3.3%	1204
50k-Rest.	0.2%	1278
Total	0.8%	864

Tabla 21. Comparación cuando las instancias son agrupadas por productos - capacidad

Por último, en la Tabla 21 mostramos los resultados cuando las instancias son agrupadas por productos-capacidad. De tablas anteriores, en donde observamos que las instancias con más productos presentan mejores resultados, así como las instancias con capacidad holgada también presentan mejores resultados, es evidente ver que las instancias agrupadas por productos-capacidad presentarán los mejores resultados. Tal como mostramos en esta tabla aquellas instancias con 50k-Holg., presentan mejores resultados respecto a los otros grupos. El tiempo de cómputo empleado para obtener estos resultados

es considerablemente mayor respecto a las demás instancias. Los mejores resultados mostrados, como se ha venido repitiendo son atribuibles a una mayor capacidad por arista y a una mejor distribución de más productos en aquellas aristas elegidas a pertenecer en la red.

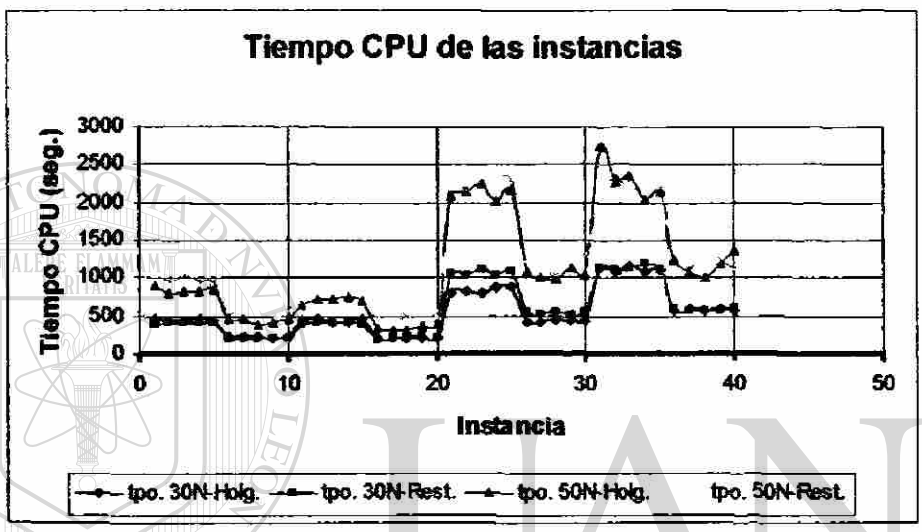


Figura 2.Gráfico de tiempo de cómputo empleado en las instancias bajo estudio

En la Figura 2 se presenta el tiempo de cómputo empleado por la metodología de búsqueda para resolver cada una de las instancias bajo estudio, los problemas fueron agrupados según la cantidad de nodos y tipo de capacidad. En el eje vertical se indica el tiempo de cómputo empleado por cada una de las instancias, en el eje horizontal se indican las instancias agrupadas por nodos-capacidad. Para diferenciar las instancias agrupadas por nodos-capacidad, se utiliza un código de colores. Las instancias de 30N-Holg. son denotadas por el color azul marino, su figura es un rombo; las instancias de 30N-Rest. se denotan por el color rosa, su figura es un cuadrado; las instancias de 50N-Holg. se diferencian de las demás por su color verde y su figura es un triángulo; por último las instancias 50N-Rest. se denotan por el color celeste y su figura son líneas paralelas.

Como se puede observar, las instancias con 30 nodos (30N-Holg. y 30N-Rest.) fueron las que presentaron los tiempos de cómputo menores respecto a las instancias de 50 nodos. Ahora bien, en ambos casos se presenta un salto abrupto a partir de la instancia 20

(eje horizontal), el tiempo de cómputo se incrementa en algunos casos al doble. Esto se debe a que a partir de la instancia 20 se agrupan las instancias con 50 productos las cuales utilizan un mayor tiempo de cómputo. De comparaciones anteriores las instancias con 50 productos presentan mejores soluciones respecto a las instancias de 30 productos, pero en cuanto al tiempo de cómputo podemos observar que emplean más recursos de cómputo, por lo tanto podemos deducir que a mayor cantidad de productos el tiempo de cómputo empleado por el algoritmo se verá incrementado.

	Tiempo CPU (seg.)		
Red	T_{min}	T_{prom}	T_{max}
30n-Holg.	200	536	1164
30n-Rest.	191	568	1179
50n-Holg.	313	1126	2738
50n-Rest.	410	1227	2456
Total	279	864	1884

Tabla 22. Tiempos de cómputo

En la Tabla 22 mostramos el tiempo de cómputo mínimo, promedio y máximo de la Figura 2. Se puede observar que el tiempo de cómputo se ve incrementado a mayor cantidad de nodos y tener una capacidad restringida. Estos tiempos se incrementan debido al aumento en la magnitud de las instancias y además que en instancias restringidas es más complicado distribuir productos por la falta de capacidad en las aristas.

CAPÍTULO 7
CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

En este último capítulo se mostrarán las conclusiones generales referentes al trabajo de tesis realizado, así como contribuciones y recomendaciones para trabajos futuros.

Como preámbulo a las conclusiones generales de esta tesis, el capítulo anterior nos muestra el comportamiento general de nuestra metodología de búsqueda y sin lugar a dudas puede ser guía para enunciar las conclusiones de este trabajo de tesis.

Como en todo procedimiento heurístico hay parámetros dentro de la metodología que tienen que ser calibrados para su más correcto desempeño. En este trabajo se determinaron tres parámetros de impacto que tuvieron que ser calibrados para la obtención de buenas soluciones. Mediante experimentación, el probar valores y determinar una combinación correcta de estos parámetros que pudiera brindar los mejores resultados fue una labor ardua que pudiera verse como un problema de optimización aparte. Así que se entregaron los valores de los parámetros para los cuales la metodología de búsqueda se comportó de manera eficiente.

En general la metodología de búsqueda entregó soluciones robustas de muy buena calidad. Hay que hacer aclaraciones en cuanto al criterio de comparación, ya que se decidió utilizar un trabajo basado en SS [14] para obtener cotas superiores de buena calidad a cada uno de los escenarios posibles al problema $DRCM_{UU}$. El trabajo empleado para obtener estas cotas superiores mostró que en general encuentra soluciones de mejor calidad que un optimizador como CPLEX, después de predeterminar un tiempo de cómputo de 8 horas. Claro como se menciona en este trabajo, en la mayoría de las ocasiones el optimizador solo encontraba la mejor solución entera después del tiempo predeterminado y no la solución óptima; razón por la cual hicimos uso del trabajo basado en SS para obtener esas cotas

superiores y utilizarlas como criterio de comparación. Además es evidente el ahorro de tiempo de cómputo empleado al usar tan novedoso e importante trabajo.

En relación al desempeño de la metodología de búsqueda propuesta, se puede notar que el número de nodos afectará enormemente los resultados arrojados. Se pudo observar que al incrementar el número de nodos en una instancia se traduce en obtener soluciones de peor calidad (caras) respecto a una instancia de menor cantidad de nodos.

Otro factor de importancia en la calidad de las soluciones obtenidas es el número de productos que circulan por la red. A mayor cantidad de productos se observó un mejor desempeño de la metodología propuesta. Sin embargo, también se ve reflejado en el tiempo de cómputo, ya que se requiere un mayor esfuerzo computacional. Los mejores resultados en las instancias con una mayor cantidad de productos puede ser atribuida a que un problema dado son aprovechadas de mejor manera las aristas elegidas para pertenecer en la red respecto a pocos productos y de esta forma pueden ser abaratados los costos.

Hay que señalar otro aspecto de importancia dentro del desempeño de la metodología de búsqueda y tiene que ver con la capacidad de las aristas en la red. A menor capacidad de las aristas, mayor es la complejidad del problema. Un problema más complejo se traduce en nuestro caso en soluciones de peor calidad (caras). Se observó, que al tener aristas con poca capacidad hace que se requiera el uso de otras aristas para transportar los requerimientos de los productos demandados, y esto hace que se incurra en pago de costos fijos por el uso de más aristas. Recordemos que los costos fijos son los más relevantes en el problema.

El desempeño de la metodología no fue afectado por el número de escenarios contemplados en un problema en particular esto lo pudimos constatar mediante experimentación.

En cuanto al tiempo de cómputo empleado es evidente que en problemas bajo incertidumbre se vuelve una variable muy importante a tratar y más según la problemática que se esté afrontando. Es por eso, que mencionar un tiempo adecuado de cómputo debe estar ligado con la aplicación que se esté estudiando. En nuestro caso podemos concluir que el tiempo de cómputo general reportado por nuestra metodología es relativamente razonable para el número de escenarios considerado.

En general en este trabajo de tesis, las soluciones entregadas por la metodología de búsqueda se encuentran a lo más, a un 0.8% de una buena cota superior del óptimo de cada escenario realizable. Con esto afirmamos que estos resultados representan un aporte en el diseño de redes bajo incertidumbre, ya que como mencionamos anteriormente este problema en particular no ha sido estudiado considerando incertidumbre en algunos de sus parámetros de entrada.

7.2 Aportaciones Científicas

En este trabajo se trató el problema de diseño de red robusta, cuya resolución se vuelve una tarea difícil y a veces intratable. El conocimiento que nosotros tuvimos acerca de este problema de diseño es que no se había considerado incertidumbre en parámetros del problema para el caso de redes con capacidades finitas, por lo que consideramos que el desarrollo e implementación de la metodología aquí presentada representa una buena contribución.

Como resultado de este trabajo de investigación se entrega una metodología de solución estructurada en programas en lenguaje C y archivos denominados *batch's* (para más detalle ver Apéndice C). Como resultado del uso de esta metodología se entrega una solución robusta al problema de $DRCM_{UU}$, que consiste en un diseño de red que se encuentra a un *P%* de la cota superior del óptimo de cada escenario, el % fue en general un 0.8%. La metodología también entrega un análisis de resultados de cada una de las iteraciones, así como del porcentaje de desviación relativa de las soluciones factibles encontradas respecto a la cota.

Los resultados del presente trabajo han sido publicados en:

- Ada M. Álvarez, Augusto Medina y Karim de Alba. A heuristic to find a P%-robust network design. *WSEAS Transactions on Circuits and Systems*, Vol. 3, pp. 2247-2253, 2004.

y presentados en:

- Augusto Medina. Diseño de una red robusta no capacitada. Seminario de Investigación. Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Junio 2003.
- Augusto Medina. Avances de un método de búsqueda para el diseño robusto de una red capacidad multiproducto. Seminario de Investigación. Facultad de Ingeniería Mecánica y Eléctrica, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, Noviembre 2003.
- Ada M. Álvarez y Augusto Medina. Robust capacitated network design problem. CORS/INFORMS Joint International Meeting, The Banff Centre, Banff, Alberta, Canadá, Mayo 16-19, 2004.

7.3 Trabajos Futuros

El presente trabajo que aquí se presentó es una extensión del problema de diseño de red general. Varias de las suposiciones hechas sobre el problema, tales como ser una red balanceada, capacitada y no orientada hacen al problema más cercano a la realidad pero a la vez mucho más difícil. La principal aportación mencionada en el punto anterior es el considerar incertidumbre en algunos de los parámetros de entrada del problema (costos variables y demanda), sin embargo no solamente estos parámetros pueden ser afectados por la incertidumbre, ya que, podría también considerase incertidumbre en el parámetro capacidad. Esto significaría un reto de estudio, por que quizás esto repercutiría en cambios de diferente naturaleza a la expuesta aquí, de tal manera que el problema antes expuesto podría ser una extensión al problema de diseño de red general.

Aquí el principal interés a futuro de este trabajo de tesis es ubicarlo en una aplicación práctica, tal como en el área de telecomunicaciones. Aunque las problemáticas de optimización en dicha área son diferentes, parte de la estructura matemática podría ser la misma para obtener un diseño de red, y optimizar algún otro criterio que se requiera. La incertidumbre también es de mucho interés en el área de telecomunicaciones afectando de manera similar que al trabajo aquí expuesto, esto abre más aún el campo de estudio en cuanto al diseño de redes y crea interesantes retos a futuro.

BIBLIOGRAFÍA

[1] I. Averbakh y O. Berman. Minmax regret median location on a network under uncertainty. *Journal on Computing,* Vol. 12, No. 2, pp. 104–110, 2000.

[2] A. Balakrishnan y T.L. Magnanti. A dual ascent procedure for large-scale uncapacitated network design. *Operations Research*, Vol. 37, No. 5, pp. 716-740, 1989.

[3] R. S. Barr, F. Glover y D. Klingman. A new optimization method for large scale fixed charge transportation problems. *Operations Research*, Vol. 29, No. 3, pp. 448-463, 1981.

[4] J. F. Benders. Partitioning procedures for solving mixed integer variables programming problems. *Numerische Methematik*, Vol. 4, pp. 238–252, 1962.

[5] J. R. Birge y F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research, Springer-Verlag, New York, EUA, 1997.

[6] N. Cobos-Zaleta. *Búsqueda Tabú para un Problema de Diseño de Red Multiproducto con Capacidad Finita en las Aristas*. Tesis de Maestría, FIME, UANL. San Nicolás de los Garza, Nuevo León, México, Julio 2004.

[7] N. Cobos-Zaleta y A. M. Álvarez. Tabu search-based algorithm for capacitated multicommodity network design problem. En *Memoria del XIV Congreso Internacional de Electrónica, Comunicaciones y Computadoras*, pp. 144-148, Veracruz, México, Febrero 2004.

[8] *CPLEX* Optimization, Inc. Incline Village, NV. ILOG CPLEX 6.6. Reference Manual, 1999.

[9] T. G. Crainic, M. Gendreau y J. M. Farvolden. A simplex-based tabu search method for capacitated network design. *Journal on Computing*, Vol. 12, No. 3, pp. 223-236, 2000.

[10] J. Darlington, C. C. Pantelides, B. Rustem y B. A. Tanyi. Decreasing the sensitivity of open-loop optimal solutions in decision making under uncertainty.

European Journal of Operational Research, Vol. 121, No. 2, pp. 343–362, 2000.

[11] M. S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. Wiley, New York, EUA, 1995.

[12] K. De Alba, A. M. Álvarez y J. L González. Un algoritmo de búsqueda para un problema de red capacitada multiproducto. En *Memorias del 3er. Encuentro Internacional de Computación, Tomo II*, pp. 105-114, Aguascalientes, México, Septiembre 2001.

[13] K. De Alba, A. M. Álvarez y J. L González-Velarde. GRASP with adaptive memory programming for finding good starting solutions to the multicommodity capacitated network design problem. En H. K. Bhargava y N. Ye, editores, *Computational Modeling and Problem Solving in the Networked World*, Capítulo 6, pp. 121-137. Kluwer, Boston, EUA, 2003.

[14] K. De Alba. *Un Procedimiento Heurístico para un Problema de Diseño de Redes Multiproducto con Capacidad Finita y Cargos Fijos*. Tesis Doctoral. FIME, UANL. San Nicolás de los Garza, Nuevo León, México, Febrero 2004.

[15] K. De Alba, A. M. Álvarez y J. L. González. Scatter search for the network design problem. Aceptado en *Annals of Operations Research*.

[16] A. Díaz, F. Glover, H. Ghaziri, J. L. González-Velarde, M. Laguna, P. Moscato, y F. Tseng. *Optimización Heurística y Redes Neuronales*. Editorial Paraninfo. Madrid, España, 1996.

[17] R. Dionne y M. Florian. Exact and approximate algorithms for optimal network design. *Networks*, Vol. 9, No. 1, pp. 37-59, 1979.

[18] G. D. Eppen, R. Kipp Martin y L. Schrage. A scenario approach to capacity planning. *Operations Research*, Vol. 37, No. 4, pp. 517-528, 1989.

[19] M. Garey y D. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, EUA, 1979.

[20] M. L. Fisher y R. Jaikumer. A decomposition algorithm for large scale vehicle routing. Reporte Técnico 78-11-05, Department of Decision Sciences, University of Pennsylvania, EUA, 1978.

[21] M. G. Florian, G. Guerin y G. Bushel. The engine scheduling problem in a railway network. *INFOR Journal*, Vol. 14, No. 1, pp. 121–138, 1976.

[22] H. N. Gabow, M. X. Goemans y D.P. Williamson. An efficient approximation algorithm for the survivable network design problem. En *Proceedings of the 3rd MPS Conference on Integer Programming and Combinatorial Optimization*, pp. 57-74. Erice, Italia, 1993.

[23] B. Gendron y T. G. Crainic. Bounding procedures for multicommodity capacitated fixed charge network design problems. Publicación CRT-96-06. Centro de Investigación del Transporte, Universidad de Montreal, Montreal, Canadá, Enero 1996.

[24] A. M. Geoffrion y G. W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, Vol. 29, No. 5, pp. 822–844, 1974.

[25] F. Glover. Tabu search - Part I. *ORSA Journal on Computing*, Vol. 1, No. 1, pp. 190-206, 1989.

[26] F. Glover y M. Laguna. *Tabu Search*. Kluwer, Norwell, MA, EUA, 1997.

[27] G. J. Gutiérrez y P. Kouvelis. A robustness approach to international sourcing. *Annals of Operations Research*, Vol. 59, No. 1, pp. 165–193, 1995.

[28] G. J. Gutiérrez, P. Kouvelis y A. A. Kurawarwala. A robustness approach to uncapacitated network design problems. *European Journal of Operational Research*, Vol. 94, No. 2, pp. 362-376, 1996.

[29] R. G. Haight, K. Ralls y A. M. Starfield. Designing species translocation strategies when population growth and future funding are uncertain. *Conservation Biology*, Vol. 14, No. 5, pp. 1298–1307, 2000.

[30] J. W. Herrmann, G. Ioannou, I. Minis y J. M. Proth. A dual ascent approach to the fixed-charge capacitated network design problem. *European Journal of Operational Research*, Vol. 95, No. 4, pp. 476-490, 1996.

[31] J. W. Herrmann. A genetic algorithm for a minimax network design problem. En *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1099-1103, Washington, D.C., EUA, Julio 1999.

[32] H. Hoang. A computational approach to the selection of an optimal network. *Management Science*, Vol. 19, No. 5, pp. 488-498, 1973.

[33] D. S. Hochbaum y A. Segev. Analysis of a flow problem with fixed charges. *Networks*, Vol. 19, No. 3, pp. 291-312, 1989.

[34] K. Holmberg y A. Migdalas. Solution methods for the discrete choice network design problem combining Lagrangean relaxation and decomposition with generation of valid inequalities. Working paper LITH-MATH/OPT-WP-1991-07. Department of Mathematics, Linkoping Institute of Techonology, Linkoping, Suecia, 1991.

[35] D. S. Johnson, J. K. Lenstra y H. G. Rinnooy. The complexity of the network design problem. *Networks*, Vol. 8, No. 2, pp. 279-285, 1978.

[36] K. Holmberg y D. Yuan. A Lagrangean approach to network design problems. *International Transactions in Operational Research*, Vol. 5, No. 6, pp. 529-539, 1998.

[37] K. Holmberg y D. Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, Vol. 48, No. 3, pp. 461-481, 2000.

[38] D. Kendrick, A. Brooke y A. Meeraus. *GAMS 5.3, A User's Guide*. The Scientific Press, San Fransisco, EUA, 1998.

[39] J. Kennington, K. Lewis, E. Olinick, A. Ortynsky y G. Spiride. Robust solutions for the WDM routing and provisioning problem: Models and algorithms. *Optical Network Magazine*, Vol. 4, No. 2, pp. 74–84, 2003.

[40] K. A. Killmer, G. Anandalingam, y S. A. Malcolm. Siting noxious facilities under uncertainty. *European Journal of Operational Research*, Vol. 133, No. 3, pp. 596–607, 2001.

[41] D. B. Khang y O. Fujiwara. Approximate solutions of capacitated fixed charge minimum cost network flow problems. *Networks*, Vol. 21, No. 6, pp. 689-704, 1991.

[42] P. Kouvelis, A. A. Kurawarwala y G. J. Gutiérrez. Algorithms for robust single and multiple period layout planning for manufacturing systems. *European Journal of Operational Research*, Vol. 63, No. 2, pp. 287–303, 1992.

[43] P. Kouvelis y G. Yu. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, Dordrecht, Holanda, 1997.

[44] M. Laguna y J. L. González-Velarde. A Benders-based heuristic for the robust capacitated internacional sourcing problem. Aceptado en *IIE Transactions*.

[45] M. Laguna. Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainty. *Management Science*, Vol. 44, No. 11, pp. S101-S110, 1997.

[46] M. Laguna, P. Lino, A. Pérez, S. Quintanilla y V. Valls. Minimizing weighted tardiness of jobs with stochastic interruptions in parallel machines. *European Journal of Operational Research*, Vol. 127, No. 2, pp. 444–457, 2000.

[47] E.L. Lawler. A procedure for computing the *k-th* best solutions to discrete optimization problems and its applications to the shortest path problem. *Management Science*, Vol. 18, No. 4, pp. 401-405, 1972.

[48] T. L. Magnanti y R. T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, Vol. 29, No. 3, pp. 464–483, 1981.

[49] T. L. Magnanti y R. T. Wong. Network design and transportation planning: Models and algorithms. *Transportation Science*, Vol. 18, No. 1, pp. 1–55, 1984.

[50] T. L. Magnanti, P. Mirchandani y R. T. Wong. Tailoring Benders decomposition for uncapacitated network design. *Mathematical Programming Study*, Vol. 26, No. 1, pp. 112–154, 1986.

[51] T. L. Magnanti, P. Mirchandani y R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, Vol. 43, No. 1, pp. 142-157, 1995.

[52] H. E. Mausser y M. Laguna. A new mixed integer formulation for the maximum regret problem. *International Transactions in Operational Research*, Vol. 5, No. 5, pp. 389– 403, 1998.

[53] H. E. Mausser y M. Laguna. A heuristic to minimax absolute regret for linear programs with interval objective function coefficients. *European Journal of Operational Research*, Vol. 117, No. 1, pp. 157-174, 1999.

[54] H. E. Mausser y M. Laguna. Minimising the maximum relative regret for linear programmes with interval objective function coefficients. *Journal of the Operational Research Society*, Vol. 50, No. 10, pp. 1063–1070, 1999.

[55] B. Melián, M. Laguna y J. A. Moreno-Pérez. Capacity expansion of fiber optic networks with WDM systems: Problem formulation and comparative analysis. *Computers and Operations Research*, Vol. 31, No. 3, pp. 461-472, 2004.

[56] M. Minoux. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, Vol. 19, No. 3, pp. 313-360, 1989.

[57] J. M. Mulvey, R. J. Vanderbei y S. A. Zenios. Robust optimization of large-scale systems. *Operations Research*, Vol. 43, No. 2, pp. 264-281, 1995.

[58] I. H. Osman y J. P. Kelly. *Metaheuristics: Theory and Applications*. Kluwer Academic Publishers, Norwell, MA, EUA, 1996.

[59] V. T. Raja y B. T. Han. A Scenario-based robust optimization model for local access network design. Aceptado en *Telecommunication Systems Journal*.

[60] R. Richardson. An optimization approach to routing aircraft. *Transportation Science*, Vol. 10, No. 1, pp. 52–71, 1976.

[61] J. Rosenhead, M. Elton y S. K. Gupta. Robustness and optimality as criteria for strategic decisions. *Operational Research Quarterly*, Vol. 23, No. 4, pp. 413–431, 1972.

[62] L. V. Snyder y M. S. Daskin. Stochastic p-robust location problems. Reporte Técnico 04T-014, Departamento de Ingeniería Industrial y Sistemas, Lehigh University, PA, EUA, Julio 2004.

[63] T. B. Trafalis, T. Mishina y B. L. Foote. An interior point multiobjective programming approach for production planning with uncertain information. *Computers & Industrial Engineering*, Vol. 37, No. 3, pp. 631–648, 1999.

[64] C. S. Yu y L. H. Li. A robust optimization model for stochastic logistic problems. *International Journal of Production Economics*, Vol. 64, No. 2, pp. 385–397, 2000.

LISTA DE TABLAS

LISTA DE FIGURAS

APÉNDICE A

DESCRIPCIÓN DE ARCHIVOS DE INSTANCIAS

A continuación detallamos un archivo de instancia de red bajo incertidumbre el cual cuenta con 4 diferentes registros. El ejemplo que se presenta es un problema con 8 nodos, 3 productos y 3 escenarios, en cada uno de los registros detallaremos el significado de los valores dados.

```
  8  12   3   3      →  Registro Tipo 1  (Nodos, Aristas, Productos, Escenarios)
0.598425                                 (Probabilidad de ocurrencia de escenario 1)
0.354331                                 (Probabilidad de ocurrencia de escenario 2)
0.047244
 1   2 76138 279
      0   0  71
      0   1  95
      0   2  71
      1   0  69
      1   1  52
      1   2  58
      2   0  53
      2   1  51
      2   2  90
 1   5 100509 235    →  Registro Tipo 2  Para cada arista: origen, destino, CF, capacidad
      0   0  67                           arista:{1,5}, costo fijo: 100509, capacidad: 235
      0   1  74
      0   2  90
      1   0  83
      1   1  73
      1   2  97
      2   0  62
      2   1  76
      2   2  82
 1   6 102617 313      Para cada producto en el escenario s que circule en la arista {1,6}
      0   0  76       →  Registro Tipo 3  (Producto, Escenario, Costo Variable)
      0   1  78            1 er. producto en escenario 2, costo variable en {1,6}=78
      0   2  69            1 er. producto en escenario 3, costo variable en {1,6}=69
      1   0  96            2 do. producto en escenario 1, costo variable en {1,6}=96
      1   1  66
      1   2  66            2 do. producto en escenario 3, costo variable en {1,6}=66
      2   0  57            3 er. producto en escenario 1, costo variable en {1,6}=57
      2   1  99
      2   2  52
 1   8 98566 318
      0   0  64
      0   1  62
      0   2  81
      1   0  64
      1   1  76
      1   2  88
      2   0  96
      2   1  65
      2   2  51
 2   6 78017 277
      0   0  56
      0   1  68
      0   2  88
```

	1	0	67
	1	1	54
	1	2	71
	2	0	81
	2	1	77
	2	2	64
2	7	79702	317
	0	0	55
	0	1	71
	0	2	91
	1	0	58
	1	1	62
	1	2	76
	2	0	89
	2	1	52
	2	2	67
3	4	82582	282
	0	0	89
	0	1	90
	0	2	57
	1	0	77
	1	1	65
	1	2	87
	2	0	81
	2	1	69
	2	2	89
3	7	92299	314
	0	0	75
	0	1	70
	0	2	62
	1	0	70
	1	1	79
	1	2	51
	2	0	88
	2	1	76
	2	2	51
4	5	79306	271
	0	0	77
	0	1	88
	0	2	63
	1	0	61
	1	1	57
	1	2	91
	2	0	74
	2	1	79
	2	2	50
4	6	99165	258
	0	0	50
	0	1	85
	0	2	80
	1	0	63
	1	1	96
	1	2	80
	2	0	67
	2	1	54
	2	2	75
6	8	86726	258
	0	0	54
	0	1	83
	0	2	89
	1	0	71
	1	1	84
	1	2	53
	2	0	70
	2	1	58
	2	2	91
7	8	78522	314
	0	0	61
	0	1	80
	0	2	63
	1	0	71

```
     1   1  58
     1   2  76
     2   0  88
     2   1  54
     2   2  69                                  Para cada producto
0    7   8          →   Registro 4              Producto, Origen, Destino
       0  61                                    Escenario, Demanda
       1  74                                    escenario: 2, demanda: 74
       2  87                                    escenario: 3, demanda: 87

1    3   1                                      Prod: 2, Origen: 3, Destino: 1
       0  72                                    escenario: 1, demanda: 72
       1  68                                    escenario: 2, demanda: 68
       2  94                                    escenario: 3, demanda: 94

2    5   6
       0  90
       1  81
       2  95
```

APÉNDICE B

GAMS/CPLEX

La formulación del problema $MCNF_s$ se guarda en un archivo de texto con extensión `*.gms (nombre.gms)`. Este archivo contiene toda la información de todos los escenarios que incurren dentro del problema original. Este archivo contiene las características del problema como nodos, productos, aristas; también se incluyen los parámetros de cada unos de los escenarios, como son costos fijos, costos variables, capacidad y demanda. El modelo matemático también es declarado dentro de este archivo, además se pueden manipular ciertas opciones del optimizador mediante un archivo cplex.opt, el cual se describe en la siguiente sección de este Apéndice.

GAMS ejecuta el archivo, crea el modelo y evoca al optimizador CPLEX 6.6 y reporta los resultados en un archivo con extensión `*.lst (nombre.lst)`.

Como complemento, hacemos notar ciertas características en este archivo *.gms para detallar algunas de las herramientas que facilitaron ya sea la ejecución o impresión de información del presente problema. Para más detalle buscar dentro del archivo *.gms el inciso detallado.

B.1. Para especificar el uso de CPLEX y utilizarlo como optimizador dentro de GAMS se declara lo siguiente:

```
Option LP = Cplex;
```

B.2. Indica el uso de una archivo de texto externo bajo la extensión *.csv, este archivo se utiliza para declarar un diseño de red que se fija al problema $MCNF_s$.

B.3. Este inciso indica el uso de dos opciones manipulables en GAMS en cuanto a detectar a un problema si es infactible o si excede cierto número de iteraciones establecidas, si se detecta alguna de estas dos opciones el proceso se aborta para evitar seguir ejecutando ese proceso y efectué trabajo demás. Se utilizan ciertos atributos del modelo para ejecutar las ideas anteriores,

nombre.modelstat=4, para indicar que un problema es infactible y luego abortar, *nombre.solvestat=2* se utiliza para indicar cierto número de iteraciones y después abortar.

B.4. Aquí se indica la impresión de información que es de utilidad para varias de nuestras rutinas, se hace impresión de la función objetivo de cada escenario, valores marginales relacionados a la restricción de capacidad para cada uno de los escenarios, también se imprime la distribución de flujo de cada uno de los escenarios.

B.5. En este inciso se hace referencia a la utilización del mismo modelo para correr cada uno de los escenarios incurridos en $MCNF_s$, y por lo tanto, generar la información requerida. Los parámetros de cada uno de los escenarios son mostrados mediante tablas en donde se declara el nombre para esos parámetros en particular, denotado con un número para indicar el escenario correspondiente. Cuando se requiere correr algún escenario solo se recargan los datos nombrando las tablas según el escenario que se vaya a optimizar.

Para más detalle mostramos a continuación un problema $MCNF_s$, el cual tiene las siguientes características: 8 nodos, 3 productos, 3 escenarios, 44 aristas.


```
$Title Problema MCNFs
Sets
i nodo          /1, 2, 3, 4, 5, 6, 7, 8/
k producto      /k1, k2, k3/
IA arcos potenciales/1.2, 2.1, 1.5, 5.1, 1.6, 6.1, 1.7, 7.1, 1.8, 8.1, 2.3, 3.2, 2.4,
4.2, 2.5, 5.2, 2.6, 6.2, 2.7, 7.2, 3.4, 4.3, 3.5, 5.3, 3.6, 6.3, 3.7, 7.3, 4.5, 5.4,
4.6, 6.4, 4.7, 7.4, 4.8, 8.4, 5.7, 7.5, 6.7, 7.6, 6.8, 8.6, 7.8, 8.7/;
alias(i,j);

Parameters
f(i,j) Costo fijo
/1.2    76138
1.5     100509
1.6     102617
1.7     89017
1.8     98566
2.3     89532
2.4     98046
2.5     92185
2.6     78017
2.7     79702
3.4     82582
3.5     82364
3.6     95832
3.7     92299
4.5     79306
4.6     99165
```

```
4.7     95997
4.8     111342
5.7     82792
6.7     84269
6.8     86726
7.8     78522/


u(i,j)  Capacidad
/1.2    279
1.5     235
1.6     313
1.7     271
1.8     318
2.3     249
2.4     292
2.5     311
2.6     277
2.7     317
3.4     282
3.5     291
3.6     264
3.7     314
4.5     271
4.6     258
4.7     239
4.8     241
5.7     317
6.7     276
6.8     258
7.8     314/;
```

```
Parameter y(i,j) arreglo existente en la actual iteración
$ondelim
$include results.csv
$offdelim
display y;
```

← **B.2**

```
Table d(i,k) Demanda de escenario actual
       k1      k2      k3
1      0       -72     0
2      0       0       0
3      0       72      0
4      0       0       0
5      0       0       90
6      0       0       -90
7      61      0       0
8      -61     0       0
;

Table c(i,j,k) Costos de transporte de escenario actual
       k1      k2      k3
2.1    71      69      53
1.2    71      69      53
5.1    67      83      62
1.5    67      83      62
6.1    76      96      57
1.6    76      96      57
7.1    71      81      57
1.7    71      81      57
8.1    64      64      96
1.8    64      64      96
3.2    67      94      66
2.3    67      94      66
4.2    98      64      90
2.4    98      64      90
5.2    91      55      55
2.5    91      55      55
6.2    56      67      81
2.6    56      67      81
```

```
7.2     55      58      89
2.7     55      58      89
4.3     89      77      81
3.4     89      77      81
5.3     98      93      54
3.5     98      93      54
6.3     97      60      72
3.6     97      60      72
7.3     75      70      88
3.7     75      70      88
5.4     77      61      74
4.5     77      61      74
6.4     50      63      67
4.6     50      63      67
7.4     78      66      94
4.7     78      66      94
8.4     83      63      72
4.8     83      63      72
7.5     93      67      65
5.7     93      67      65
7.6     67      98      60
6.7     67      98      60
8.6     54      71      70
6.8     54      71      70
8.7     61      71      88
7.8     61      71      88
;

Table d2(i,k) Demanda de escenario actual
        k1      k2      k3
1       0       -68     0
2       0       0       0
3       0       68      0
4       0       0       0
5       0       0       81
6       0       0       -81
7       74      0       0
8       -74     0       0
;

Table c2(i,j,k) Costos de transporte de escenario actual
        k1      k2      k3
2.1     95      52      51
1.2     95      52      51
5.1     74      73      76
1.5     74      73      76
6.1     78      66      99
1.6     78      66      99
7.1     63      98      64
1.7     63      98      64
8.1     62      76      65
1.8     62      76      65
3.2     69      71      78
2.3     69      71      78
4.2     55      68      77
2.4     55      68      77
5.2     90      71      68
2.5     90      71      68
6.2     68      54      77
2.6     68      54      77
7.2     71      62      52
2.7     71      62      52
4.3     90      65      69
3.4     90      65      69
5.3     59      72      85
3.5     59      72      85
6.3     50      90      91
3.6     50      90      91
7.3     70      79      76
3.7     70      79      76
5.4     88      57      79
```

```
4.5     88      57      79
6.4     85      96      54
4.6     85      96      54
7.4     54      90      88
4.7     54      90      88
8.4     53      51      79
4.8     53      51      79
7.5     62      67      72
5.7     62      67      72
7.6     79      91      60
6.7     79      91      60
8.6     83      84      58
6.8     83      84      58
8.7     80      58      54
7.8     80      58      54
;

Table d3(i,k) Demanda de escenario actual
        k1      k2      k3
1       0       -94     0
2       0       0       0
3       0       94      0
4       0       0       0
5       0       0       95
6       0       0       -95
7       87      0       0
8       -87     0       0
;

Table c3(i,j,k) Costos de transporte de escenario actual
        k1      k2      k3
2.1     71      58      90
1.2     71      58      90
5.1     90      97      82
1.5     90      97      82
6.1     69      66      52
1.6     69      66      52
7.1     96      94      61
1.7     96      94      61
8.1     81      88      51
1.8     81      88      51
3.2     50      52      99
2.3     50      52      99
4.2     78      72      67
2.4     78      72      67
5.2     67      99      95
2.5     67      99      95
6.2     88      71      64
2.6     88      71      64
7.2     91      76      67
2.7     91      76      67
4.3     57      87      89
3.4     57      87      89
5.3     53      69      91
3.5     53      69      91
6.3     63      77      99
3.6     63      77      99
7.3     62      51      51
3.7     62      51      51
5.4     63      91      50
4.5     63      91      50
6.4     80      80      75
4.6     80      80      75
7.4     67      94      98
4.7     67      94      98
8.4     93      99      74
4.8     93      99      74
7.5     93      90      64
5.7     93      90      64
7.6     94      80      79
6.7     94      80      79
```

```
8.6     89      53      91
6.8     89      53      91
8.7     63      76      69
7.8     63      76      69
;

variables
        x(i,j,k)        flujo en el arco
        z               ganancia;

Positive variable x

Equations
            costo           define la funcion objetivo
            sa1             restriccion de no violar la capacidad de cada arco
            sa2             restriccion de conservacion de flujo

costo  ..   z =e= sum ( (i,j),f(i,j)*y(i,j) ) + sum( (k,i,j), c(i,j,k) * x(i,j,k) ) ;

sa1(i,j)$(ORD(i) LT ORD(j)) .. sum (k, x(i,j,k) + x(j,i,k) ) =l= u(i,j)*y(i,j) ;

sa2(k,i) .. sum(j$IA(i,j), x(i,j,k)) - (sum(j$IA(j,i), x(j,i,k))) =e= d(i,k) ;
```

```
Model DRCM /all/;
option limcol=0;
option limrow=0;

option lp = cplex;
DRCM.optfile=1;

d(i,k)=d(i,k);
c(i,j,k)=c(i,j,k);
option BRATIO=1;
Solve DRCM using lp minimizing z;
*Display z.l,X.l;
*display sa1.m;
```

B.1

```
if (DRCM.solvestat=2,
abort$(DRCM.solvestat = 2) "Model is infeasible";
);

if (DRCM.modelstat=4,
abort$(DRCM.modelstat = 4) "Model is infeasible";
);
```

B.3

```
file objetivo1/objetivo1.txt/;
  put objetivo1 ;
  put z.l;

file marginal1 /marginal1.out/;
  put marginal1
  set ijseps(i,j);
  ijseps(i,j)$sa1.m(i,j)= yes$(sa1.m(i,j)=0);
  LOOP ((i,j)$(not ijseps(i,j)), put I.tl, J.tl, sa1.m(i,j),''/);

file flujos1 /flujos1.out/;
  put flujos1
  put @1#1;
  LOOP((I,J,K)$x.l(i,j,k), put@1 I.tl, J.tl, K.tl, x.l(i,j,k),''/);
```

B.4

```
d(i,k)=d2(i,k);
c(i,j,k)=c2(i,j,k);
```

B.5

```
option BRATIO=1;

Solve DRCM using lp minimizing z;
*Display z.l,X.l;
*display sal.m;


if (DRCM.solvestat=2,
abort$(DRCM.solvestat = 2) "Model is infeasible";
);

if (DRCM.modelstat=4,
abort$(DRCM.modelstat = 4) "Model is infeasible";
);

file objetivo2/objetivo2.txt/;
  put objetivo2 ;
  put z.l;

file marginal2 /marginal2.out/;
  put marginal2
  set ijseps(i,j);
  ijseps(i,j)$sal.m(i,j)= yes$(sal.m(i,j)=0);
  LOOP ((i,j)$(not ijseps(i,j)), put I.tl, J.tl, sal.m(i,j),''/);

file flujos2 /flujos2.out/;
  put flujos2
  put @1#1;
  LOOP((I,J,K)$x.l(i,j,k), put@1 I.tl, J.tl, K.tl, x.l(i,j,k),''/);


d(i,k)=d3(i,k);
c(i,j,k)=c3(i,j,k);
option BRATIO=1;

Solve DRCM using lp minimizing z;
*Display z.l,X.l;
*display sal.m;


if (DRCM.solvestat=2,
abort$(DRCM.solvestat = 2) "Model is infeasible";
);

if (DRCM.modelstat=4,
abort$(DRCM.modelstat = 4) "Model is infeasible";
);

file objetivo3/objetivo3.txt/;
  put objetivo3 ;
  put z.l;

file marginal3 /marginal3.out/;
  put marginal3
  set ijseps(i,j);
  ijseps(i,j)$sal.m(i,j)= yes$(sal.m(i,j)=0);
  LOOP ((i,j)$(not ijseps(i,j)), put I.tl, J.tl, sal.m(i,j),''/);

file flujos3 /flujos3.out/;
  put flujos3
  put @1#1;
  LOOP((I,J,K)$x.l(i,j,k), put@1 I.tl, J.tl, K.tl, x.l(i,j,k),''/);
```

Archivo cplex.opt

Como anteriormente se declaró utilizamos el optimizador CPLEX 6.6 para resolver los problemas $MCNF_s$, de este optimizador son modificadas ciertas opciones para facilitar la ejecución del mismo.

El archivo *cplex.opt* permite la manipulación de esas opciones utilizadas por el optimizador CPLEX. Este es un archivo de texto en donde se declaran los parámetros del optimizador que se pretenden cambiar, a continuación mencionamos los parámetros de los cuales hacemos uso, valores a los cuales los predeterminamos y una pequeña descripción de ellos:

- `simdisplay:` esta opción permite manipular la impresión en pantalla de información de una corrida de un problema, un valor de 0 indica que no reporta información en pantalla hasta que la solución óptima es obtenida.
- `iterlim:` permite manipular el número de iteraciones permitidas para el proceso de solución del problema $MCNF_s$.
- `reslim:` permite estipular el tiempo permitido para resolver el problema $MCNF_s$, el tiempo se especifica en tiempo CPU (segundos).
- `bratio:` permite utilizar o no bases anteriores. Un valor de 1 origina que GAMS indique a CPLEX el uso de bases anteriores.

APÉNDICE C

RUTINAS Y MECANISMO DE EJECUCIÓN DE LA METODOLOGÍA DE BÚSQUEDA

Para detallar el uso de la metodología de búsqueda se estructuro el contenido en tablas como se muestra a continuación.

Se definen archivos *batch* para ejecutar la metodología de búsqueda. Un archivo *batch* es un archivo de texto que es convertido a ejecutable en plataforma UNIX mediante el comando *chmod 775*. El contenido de este archivo es ya sea un programa u optimizador que se desee ejecutar en una línea de comando, cada línea definida en el archivo *batch* es una tarea a realizar.

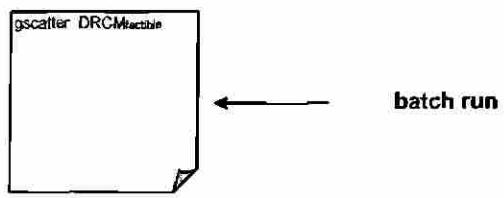
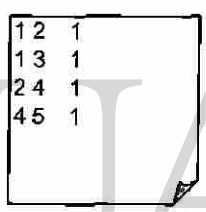
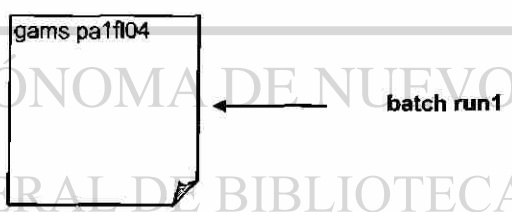
Las rutinas implementadas también son definidas así como algunas de sus características más importantes, como nombre, archivo ejecutable, que hacen, entradas necesarias para ejecutar el programa, y salidas de cada programa.

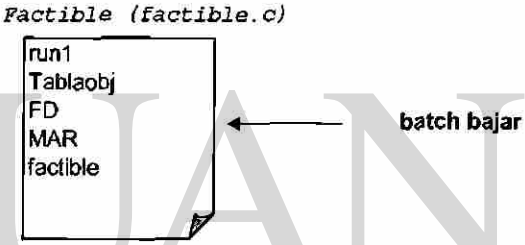
PROGRAMA	DESCRIPCIÓN
GENERADORprob.c	Este programa genera N instancias bajo incertidumbre. Las entradas de este programa son nodos, productos y escenarios. Salidas: Se crean N instancias según la agrupación de problemas que tenemos con las diferentes características expuestas. Separa en carpetas los problemas según su capacidad holgada o restringida. En cada carpeta puedo tener así mismo varios problemas de ese mismo tipo. La carpeta con una instancia específica (por ejemplo un palf101), contiene: El problema original bajo incertidumbre. Ejecutables y batch´s a utilizar para aplicar la metodología de búsqueda.
FLUJOS.c	Esta rutina hace el cálculo de la siguiente operación: $a_{ij} = \frac{F_{ij}}{\frac{\sum_{s \in S} \sum_{k \in K} x^k_{ij}(s) + x^k_{ji}(s)}{\lvert S \rvert}} \quad \forall \{i,j\} \in A.$ Las entradas que necesita este programa son los archivos flujos$S$.out generados por los sub-problemas $MCNF_s$ y archivo de costosfijos.txt. La salida es un archivo denominado FLUJOS.txt.
Dualfactor.c	Esta rutina hace el cálculo del beneficio relativo (r_{ij}), mediante el siguiente criterio: Donde a_{ij} se calcula por el programa anterior. $r_{ij} = \begin{cases} \frac{E(\pi_{ij})}{a_{ij}} & \text{si } E(\pi_{ij}) < 0 \\ a_{ij} & \text{si } E(\pi_{ij}) = 0 \end{cases}$ Las entradas de este programas son marginales.out, que es una salida de los sub-problemas $MCNF_s$, y el archivo FLUJOS.txt generado por el programa FLUJOS.c Efectuado el cálculo se acomoda en orden de mayor a menor. La salida es una archivo denominado qq.txt, este contiene al listado de r_{ij}.

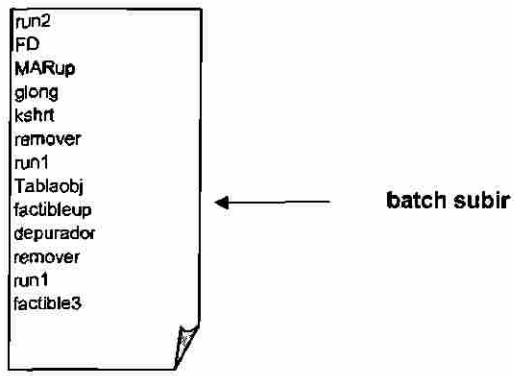
PROGRAMA	DESCRIPCIÓN
qd3.c	Esta rutina elimina las aristas mencionadas en el archivo qq.txt, las elimina del archivo results1.txt, results.csv, Todos.txt, Todos.csv, y cada arista que va eliminando se almacena en Removidos.txt. Archivo de entrada qq.txt. Archivo de salida Removidos.txt.
factible.c	Esta rutina hace una exploración sobre las primeras "⌋" aristas declaradas en qq.txt, elimina de las "⌋" aristas solo aquellas que no provoquen infactibilidad al problema. Archivos de entrada objetivosS.txt, un batch denominado run1 y programa Tablaobj.c, y qd3.c.
Tablaobj.c	Este programa almacena los valores de las funciones objetivo de cada uno de los diseños que se van obteniendo en cada una de las iteraciones. Además obtiene la desviación relativa respecto a una cota superior de buena calidad. Archivos de entrada objetivoS.txt. Archivo salida almacenobj.txt.
dualfactorup.c	Rutina parecida a dualfactor.c, solo que esta hace comparaciones con otros archivos.
kshrt (executable)	Este ejecutable carga kshort.txt, y determina las k-rutas más cortas para aquellos productos que perdieron factibilidad. Las k-rutas más cortas se almacenan para su análisis, esto es, al principio solo se utilizará la 1er ruta más corta para cada producto que perdió factibilidad. Las otras rutas se conservan para su posterior posible uso, esto es, si no se recupera factibidad con este primer conjunto de rutas, entonces se hace uso de siguiente mejor conjunto de rutas. Se almacenan 5 conjuntos de rutas. Las rutas son separadas en aristas, y agragadas en results1.txt, y results.csv. Entradas del programa kshort.txt. Salidas del programa results1.txt, results.csv.

PROGRAMA	DESCRIPCIÓN
glong.c	Rutina que se utiliza al momento de recuperar factibilidad en un problema que la perdió. Este programa prepara el archivo denominado kshort.txt. Este archivo aprovecha la salida de dualfactorup.c, la cual es qq.txt, el contenido de este es el valor de los r_{ij}. La información de los r_{ij} se utilizará para calcular la longitud de una arista. Se toma en consideración dentro del cálculo a las capacidades de una arista de la siguiente manera: $$l_{ij} = \frac{r_{ij}}{u_{ij}} \qquad \forall \{i,j\} \in A.$$ Una vez hecho el cálculo, se prepara el archivo kshort.txt, el contenido de este archivo especifica solo las arista que no han sido utilizadas en el problema. No se incluyen aristas que se encuentren en el actual diseño de red, tampoco se encuentra alguna arista que se mencione en el archivo Removidos.txt. Determinar a que productos se provocó infactibilidad e indicar la trayectoria que sigue su demanda y reportarla en el contenido de este archivo. Archivos entrada son qq.txt, capacidades.txt, demandas.txt. El archivo salida es kshort.txt.
factibleup.c	Esta rutina comprueba que se haya recuperado factibilidad al paso anterior de agregar rutas. Si se recupera la factibilidad se sale del programa, si NO se recupera la factibilidad entonces asigna la segunda mejor ruta, esto lo puede hacer hasta 5 veces en caso de no cumplirse la recuperación de factibilidad. Entradas, un batch denominado run1, objetivos.txt.
factible3.c	Programa que pregunta si ante la depuración se recupera la factibilidad. Si se recupera la factibilidad al hacer la depuración entonces se sale del programa, si NO se recupera la factibilidad entonces se evoca al programa nuevo.c que definiremos mas adelante.

PROGRAMA	DESCRIPCIÓN
depurador.c	Rutina que depura las aristas agregadas. A las aristas agregadas se les hace un análisis para determinar que arista es esencial y cual no. Las aristas esenciales son las que transportan flujo en todos los escenarios del problema. La aristas esenciales quedan fijas dentro del actual diseño robusto, y las que no cumplen esta condicion son eliminadas, tanto de results1.txt como de results.csv. Archivos entrada agregados.txt Archivos salida results1.txt, results.csv.
nuevo.c	Este programa reasigna aristas del último listado agregados.txt, esta rutina se activa en caso de que no se haya recuperado la factibilidad al momento de hacer la depuración. Por lo tanto de las aristas que en dos pasos anteriores se definieron como no esenciales, de estas nececitamos agregas algunas de ellas, cuales?. Agregar aquellas aristas que pertenecen a alguno de los vértices de las trayectorias de los productos que perdieron factibilidad. Archivos entrada agregados.txt, demandas.txt. Archivos salida, results1.txt, results.csv.
reintegracion.c	Esta rutina inserta las aristas del archivo Removidos.txt a los archivos Todos.txt y Todos.csv, después de W iteraciones.

ARCHIVOS BATCH	CONTENIDO Y QUE HACE
run	Batch que determina una solución factible inicial a un problema dado. Se indica el ejecutable del Scatter Search seguido del problema $DRCM_{factible}$  La entrada a este batch es un archivo bajo el formato de instancia de red que se definió en el punto de instancias creadas. La salida es un archivo con un diseño de red inicial denominado results1.txt y results.csv. Se indica el arco (i,j), se indica 0 ó 1 si se contempla un arco o no dentro del diseño. Se indica el ejecutable del Scatter 
run1	Batch que contiene en formato *.gms el problema $MCNF_s$. Evocar al modelador GAMS seguido del nombre del problema.  El problema $MCNF_s$ evoca al 1^{er} diseño factible inicial y a las posteriores modificaciones de results1.txt y results.csv. Este archivo ejecuta los s sub-problemas de $MCNF_s$, las salidas que produce son: obtjetivo*S*.txt, flujos*S*.out, marginales*S*.out.
run2	Este batch se utiliza para el PASO 3.2, recuperación de factibilidad. Este batch contiene un archivo en formato *.gms. Contiene un archivo de características similares a $MCNF_s$, solo que evoca a un diseño Y^{aux}. Este archivo evoca a un archivo denominado Todos.txt y Todos.csv (Y^{aux}). Este archivo ejecuta los *s* sub-problemas y las salidas que produce son: marginales*S*.out, y flujos*S*.out.

R1……Rs	Archivos batch que evocan la rutina del Scatter Search. Determina una cota superior para cada escenario en el problema original $DRCM_{UU}$.
Runner	Evoca a los archivos R1……Rs.
BATCH para aplicar la metodología	El batch *bajar* contiene el PASO 2 y 3.1 de la metodología de búsqueda. Bajar: este batch contiene los ejecutables de los siguientes programas. **(Archivo ejecutable - Nombre del programa)** *run1* *Tablaobj (Tablaobj.c)* *FD (FLUJOS.c)* *MAR (dualfactor.c)* *Factible (factible.c)*  El batch *subir* contiene el PASO 3.2 de la metodología de búsqueda. *Subir:* este batch contiene los ejecutables de los siguientes programas. *run2* *FD (FLUJOS.c)* *MARup (dualfactorup2.c)* *glong (glong.c)* *kshrt (execute)* *remover (batch que limpia entorno)* *run1* *Tablaobj (Tablaobj.c)* *factibleup (factibleup.c)* *depurador (depurador.c)* *remover* *run1* *factible3 (factible3.c)*



Nvueltas: este batch ejecuta la metodología de búsqueda. En contenido de este batch es:

run

bajar

subir

bajar

subir

…

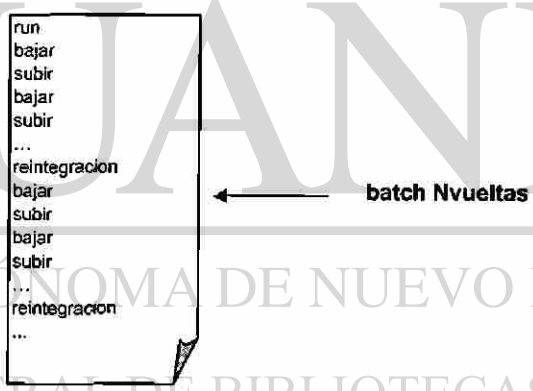
reintegración

bajar

subir

bajar

subir



…

reintegración

bajar

subir

bajar

subir

…

Este batch se ejecuta según el número de iteraciones que se especifique.

APÉNDICE D

ESTUDIO DE LA FASE DE DEPURACIÓN

Este experimento se hizo con la finalidad de justificar el uso ó no uso de la fase de depuración dentro de la metodología de búsqueda. Para ello se tomó una muestra significativa de las instancias generadas para el estudio computacional (50% del total) con las cuales se realizó el siguiente estudio:

- Se aplicó la metodología sin incluir la fase de depuración.
- Se aplicó la metodología incluyendo la fase de depuración.

El análisis de resultados se muestra en la Tabla 20 que se muestra a continuación:

	Aplicando Depuración				Sin aplicar Depuración			
Instancias	**Inicial**	**Final**	**Mejora**	**Tiempo**	**Inicial**	**Final**	**Mejora**	**Tiempo**
30 N	19.7%	-1.4%	21.1%	478	19.7%	0.8%	18.9%	690
50 N	24.5%	3.2%	21.3%	993	24.5%	3.7%	20.8%	1478
General	22.1%	0.9%	21.2%	735	22.1%	2.3%	19.8%	1084

Tabla 23. Estudio de la fase de depuración

El estudio se centró en incluir ó no la fase de depuración dentro de la metodología, y comparar sus resultados. Los rubros que se indican en la tabla son los siguientes: *Instancias*, según la cantidad de nodos (30 ó 50); *Inicial*, indica el porcentaje de desviación relativa del costo total de la solución inicial de la que se parte, respecto a una cota superior del costo óptimo; *Final*, indica el porcentaje de desviación relativa de la solución que entrega el algoritmo respecto a la cota superior (los valores con signo negativo reportados en este rubro indican una mejora a favor de la metodología con respecto a la cota superior). En este rubro conviene aclarar que para cada una de las instancias el porcentaje de desviación relativa es el mayor entre todos los escenarios posibles en una instancia en particular. *Mejora*, nos indica el porcentaje de mejora desde la solución inicial

hasta la solución robusta obtenida; por último el *Tiempo*, que nos indica el tiempo CPU en segundos que la metodología de búsqueda empleó para obtener la solución robusta.

Según los resultados obtenidos en este experimento, cuando se incluye la fase de depuración se obtienen mejores soluciones respecto a cuando no se incluye. El porcentaje de desviación relativa general que arrojó la metodología cuando se incluye la fase de depuración fue de 0.9% y cuando no 2.3%.

Lo más significativo de este estudio se observa en el tiempo de cómputo, ya que cuando no se considera la fase de depuración el tiempo se incrementa considerablemente según lo muestra la tabla comparativa. Este aumento en el tiempo de cómputo se debe a que al omitir la fase de depuración el proceso siguiente sería eliminación de aristas según el Paso 3.1. Este proceso es lento y en caso de tener un diseño con muchas aristas (siendo algunas de ellas no necesarias para su uso) hace que el proceso de eliminación de aristas tarde al resolver por algunas iteraciones problemas lineales (tantos como escenarios existan) y por lo tanto se incremente dicho tiempo de cómputo.

APÉNDICE E

GRÁFICOS DE DESEMPEÑO DE CADA INSTANCIA EVALUADA

En este Apéndice mostramos el desempeño de cada una de las instancias evaluadas y analizadas según su rubro. Cada gráfico muestra en la parte superior el rubro por el cual están acomodadas las instancias, en el eje horizontal se indican las instancias, en el eje vertical se indican los porcentajes de desviación relativa respecto a una cota superior. En la parte inferior del gráfico se muestra en un recuadro como se señala el porcentaje de desviación relativa de la solución inicial y final de una instancia (en color azul marino y amarillo respectivamente). La graficación muestra el porcentaje de desviación relativa de la solución inicial y final de cada instancia acomodada según su rubro. Las instancias son acomodadas según cantidad de nodos (30, 50), según capacidad (holgada, restringida), cantidad de productos (30, 50), numero de escenarios (10, 20) y por densidad (50%, 75%).

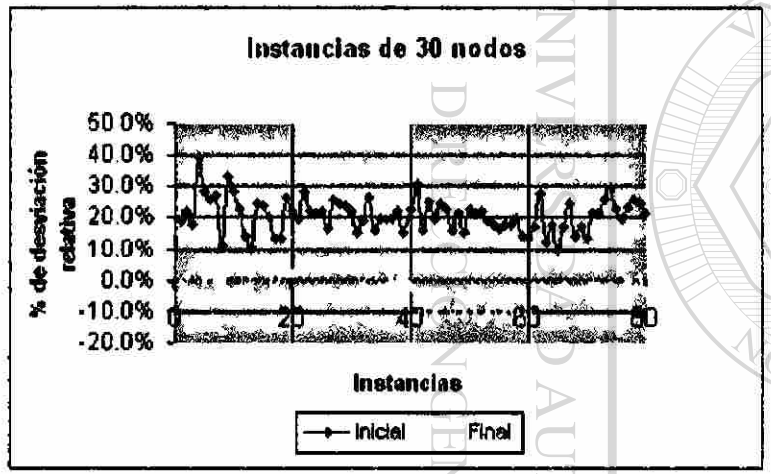


Figura 3. Desempeño de la metodología de búsqueda en instancias de 30 nodos

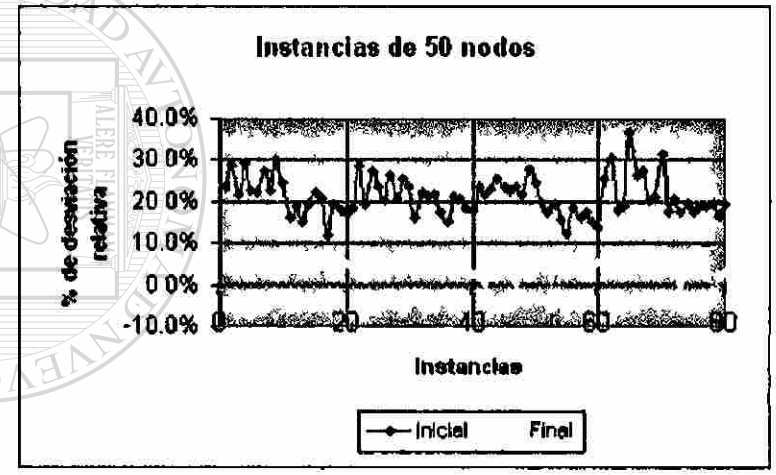


Figura 4. Desempeño de la metodología de búsqueda en instancias de 50 nodos

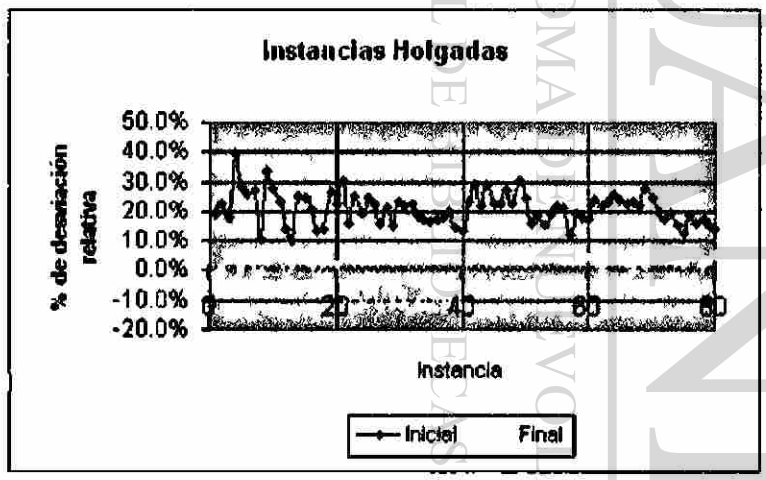


Figura 5. Desempeño de la metodología de búsqueda en instancias con capacidad holgada

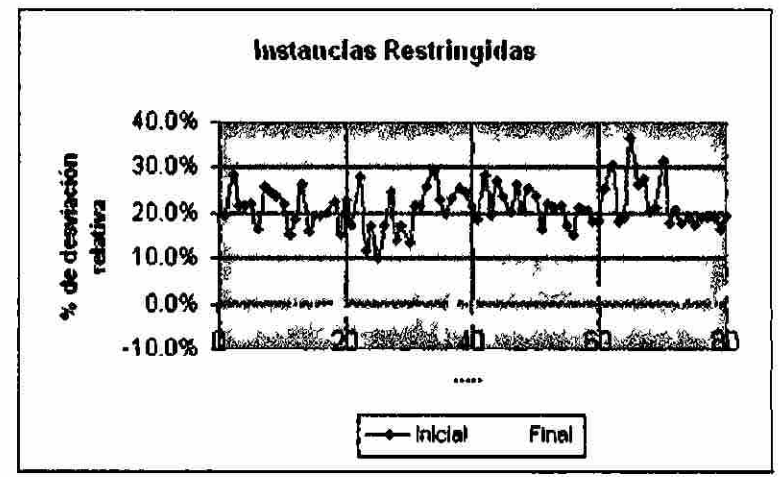


Figura 6. Desempeño de la metodología de búsqueda en instancias con capacidad restringida

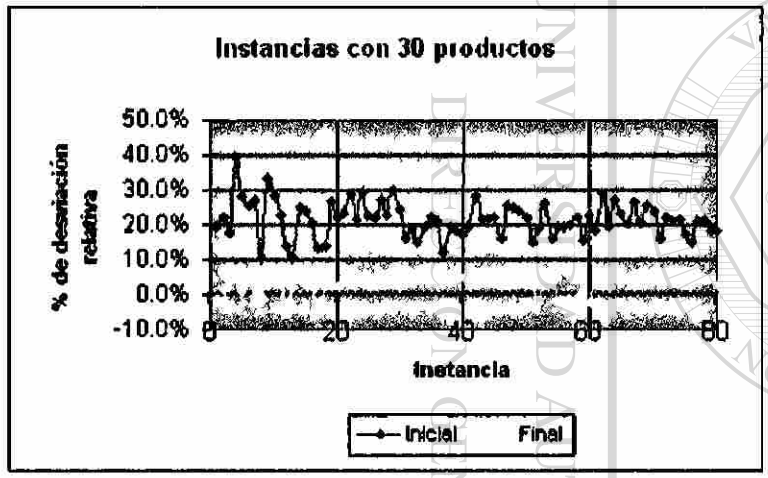


Figura 7. Desempeño de la metodología de búsqueda en instancias con 30 productos

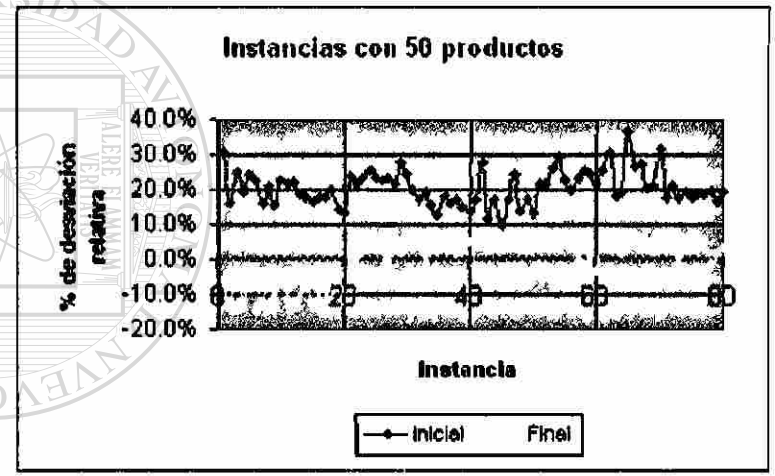


Figura 8. Desempeño de la metodología de búsqueda en instancias con 50 productos

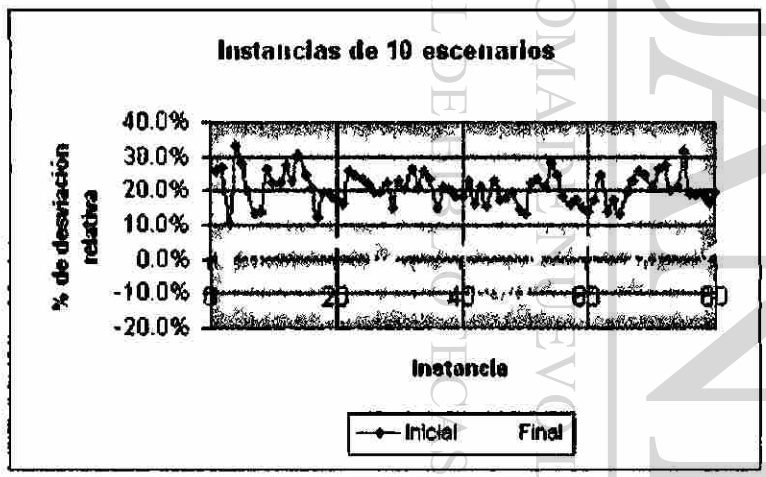


Figura 9. Desempeño de la metodología de búsqueda en instancias con 10 escenarios

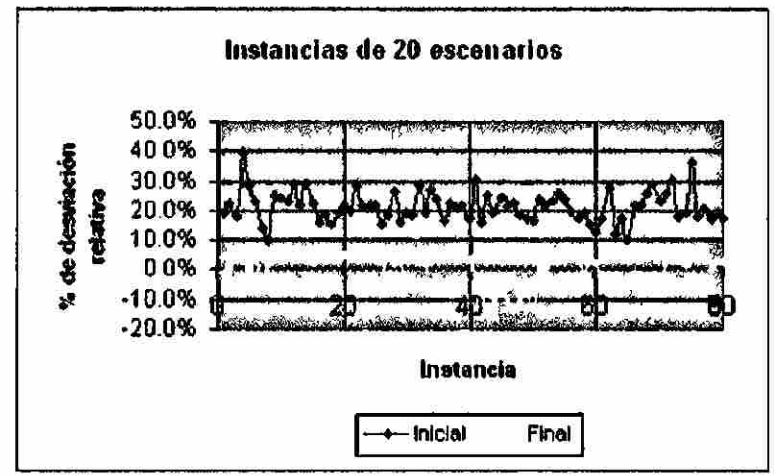


Figura 10. Desempeño de la metodología de búsqueda en instancias con 20 escenarios

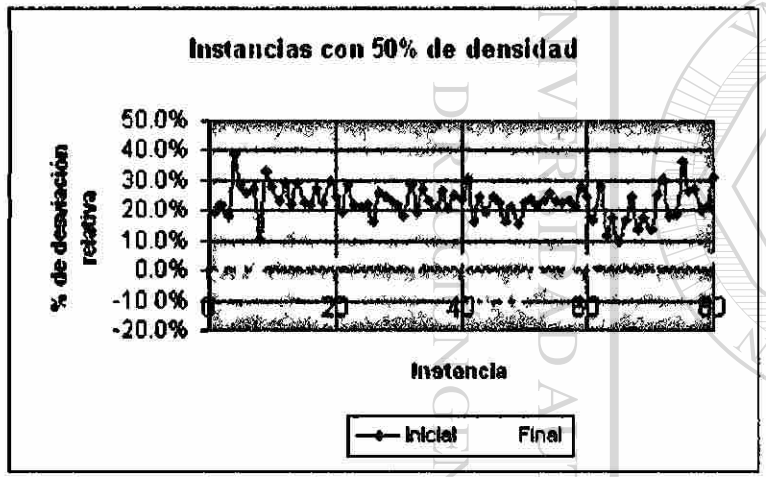


Figura 11. Desempeño de la metodología de búsqueda en instancias con 50% de densidad

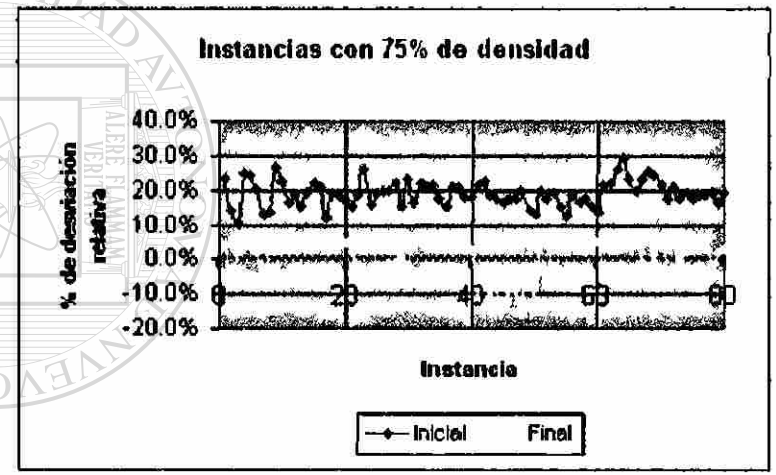


Figura 12. Desempeño de la metodología de búsqueda en instancias con 75% de densidad

FICHA AUTOBIOGRÁFICA

José Florentino Augusto Medina Jacobo

Candidato para el grado de Maestro en Ciencias en Ingeniería de Sistemas
Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis

UN ENFOQUE ROBUSTO A UN PROBLEMA DE DISEÑO DE RED MULTIPRODUCTO CON INCERTIDUMBRE EN PARÁMETROS DE ENTRADA

Nacido en la ciudad de Monterrey, Nuevo León. Hijo único del Sr. José de Jesús Medina Ortiz y la Sra. Nemecia Jacobo Ramírez. Graduado en la Facultad de Ciencias Químicas en la Universidad Autónoma de Nuevo León (1995-2000) como Ingeniero Industrial Administrador. Se desempeñó como Auxiliar de Producción en la empresa FORMET, S.A. de C.V. (1999-2000), también se desempeño como Consultor de ISO-9000 en la empresa Calefacción Eléctrica, S.A. de C.V. (2001). En Febrero del 2002 ingresó a la Maestría en Ciencias en Ingeniería de Sistemas. Obtuvo una beca como Asistente de Investigación por el proyecto 36669-A de CONACYT para realizar sus estudios de maestría en la División de Posgrado en Ingeniería de Sistemas, de la Facultad de Ingeniería Mecánica y Eléctrica en la Universidad Autónoma de Nuevo León.