UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
DIVISIÓN DE ESTUDIOS DE POSGRADO

UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE

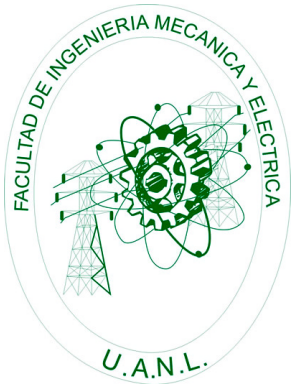
Por

Fernando Pérez González

Como requisito parcial para obtener el Grado de MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS

Ciudad Universitaria. Diciembre, 2005

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA
DIVISIÓN DE ESTUDIOS DE POSGRADO



UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE

Por

Fernando Pérez González

Como requisito parcial para obtener el Grado de MAESTRO EN CIENCIAS EN INGENIERÍA DE SISTEMAS

Ciudad Universitaria. Diciembre, 2005

Universidad **A**utónoma de **N**uevo **L**eón

Facultad de **I**ngeniería **M**ecánica y **E**léctrica

División de **E**studios de **P**osgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE**"**, realizada por el alumno Fernando Pérez González, con número de matrícula 915821 sea aceptada para su defensa como opción al grado de Maestro en Ciencias en Ingeniería de Sistemas.

El Comité de Tesis

_____

Dra. Ada M. Álvarez Socarrás

Asesor

_____ _____

Dr. Karim De Alba Romenus — Revisor

Dr. Arturo Berrones Santos — Revisor

Vo. Bo.

_____

Dr. G. Alan Castillo Rodríguez

Subdirector

División de Estudios de Posgrado

Ciudad Universitaria. Diciembre, 2005

DEDICATORIA

Quiero dedicar el presente trabajo de tesis a todos aquellos que de una u otro forma estuvieron siempre conmigo: familia, maestros y amigos; fueron ellos quienes me dieron la fuerza de luchar y el gusto de seguir siempre adelante.

A mis padres Fernando Pérez Chávez y Julia Adela González Dávalos, quienes me dieron cuanto estuvo a su alcance y hoy mis logros les pertenecen. Siempre han estado en mi corazón sin importar cuán lejos pudieran estar.

A mi hermano Arturo Pérez González que ha sido un gran compañero y María Julia, mi pequeña sobrina que llevo en mi pensamiento siempre.

A mis amigos entrañables Nancy Garza, Fernando Reyes, Israel Cano, Oswaldo Moreno y Miguel Mata, así como a los demás compañeros de otros semestres. Qué grata fue mi estancia en la maestría gracias a todos ellos.

A mis maestros que supieron cómo alegrar cada instante de permanencia en el programa de postgrado. Creo que no hubo un día que hubiera sido difícil porque estaban ellos ahí para aligerar cualquier carga.

Y por último, quiero dedicar esta tesis a mi novia que amo profundamente, Beatriz Gabriela Garza Montoya, quien siempre ha creído en mí, que me ha apoyado y que sigue conmigo aquí a mi lado, como lo ha hecho hace tanto y como sé lo seguirá haciendo por mucho tiempo más.

AGRADECIMIENTOS

Quiero agradecer sinceramente a mi asesora de tesis, la Dra. Ada M. Álvarez Socarrás por haber dedicado tanto tiempo y afecto a mi persona, siempre me supe a salvo con ella, una persona de lo mejor.

A mi asesor Karim De Alba Romenus, que me supo guiar y enseñar sobre temas diversos de profundidad científica, que me enseñó aspectos nuevos del mundo y frases siempre a tono con la situación.

A mi asesor Arturo Berrones Santos, una persona tan agradable y sincera, dando siempre aportaciones de gran valor y apoyo para la realización de esta tesis.

A todos los maestros del PISIS, por haber sido no solo mentores, sino amigos de verdad, por ayudarme a crecer en ámbitos profesional y personal.

Al CONACYT por haberme dado esta increíble oportunidad de aprendizaje, de crecimiento, de esperanza., de encontrar personas tan valiosas. Hallé tanto en este lugar, que no quisiera dejarlo nunca.

A mi familia, a mis compañeros y maestros que hicieron de esté período de mi vida algo tan maravilloso. A todos ustedes gracias.

RESUMEN

Fernando Pérez González
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:

UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE

Número de páginas: 107

OBJETIVO Y MÉTODO DE ESTUDIO

El presente trabajo aborda un problema de diseño de red capacitada multiproducto con incertidumbre en algunos de sus parámetros de entrada, como lo son las demandas y costos variables de sus productos.

La versión determinista de este problema, en donde no se incluye incertidumbre, se modela como un programa no lineal entero mixto (compuesto por variables enteras y continuas) y es un problema clasificado como combinatorio y de complejidad no-polinomial, lo que significa que no hay algoritmos exactos para resolverlo en un tiempo razonable, incluso para instancias relativamente pequeñas.

La versión que incluye incertidumbre es aún más compleja y para poder tratarse se hacen necesarias metodologías heurísticas que puedan encontrar soluciones buenas en un tiempo de cómputo aceptable, y técnicas del tipo de Optimización Robusta o Estocástica para manejar la incertidumbre.

El objetivo principal de nuestro trabajo es desarrollar e implementar una metodología que entregue una solución robusta (factible y relativamente buena sin importar el escenario en que se evalúe), al problema de diseño de red capacitada multiproducto bajo incertidumbre

En la presente tesis se desarrolla una metodología de solución al problema en cuestión, basada en la metaheurística GRASP y empleando técnicas de Optimización Robusta, la cual trabaja con escenarios para modelar la incertidumbre del problema.

CONTRIBUCIONES Y CONCLUSIONES

Como resultado del presente trabajo se entrega un sistema computacional basado en los procedimientos propuestos al problema de diseño de red capacitada multiproducto bajo incertidumbre. El diseño entregado como resultado especifica los costos de la solución en un formato que permite observar los costos fijos y variables incurridos en el diseño, así como los costos según el escenario a tratar.

La validación de los resultados se realiza mediante comparación con cotas de buena calidad y con un trabajo previo que resuelve el problema que tratamos. Nuestros resultados fueron generalmente superiores respecto a las comparaciones.

La contribución científica del presente trabajo se sustenta en el hecho de aplicar un procedimiento heurístico que aporta soluciones de buena calidad en tiempo aceptable a un problema para el cual no hay metodología de solución exacta en tiempo razonable hasta la fecha y que no ha sido suficientemente tratado en la literatura especializada.

Asesor:

_____

Dra. Ada M. Álvarez Socarras

ÍNDICE

UNA METODOLOGÍA DE SOLUCIÓN BASADA EN LA METAHEURÍSTICA GRASP PARA EL PROBLEMA DE DISEÑO DE RED CON INCERTIDUMBRE

CAPÍTULO 1. INTRODUCCIÓN

1.1 Descripción y Relevancia del Problema.

Los problemas de diseño de red han sido muy estudiados debido a su importancia en el área de Investigación de Operaciones y su utilidad en aplicaciones diversas, tales como inversiones de capital, toma de decisiones para planeación de transportación, ruteo de vehículos, planeación de flotas, diseño de sistemas para manejo de material, diseño de redes de telecomunicaciones, localización de plantas y diseño de sistemas de distribución (Magnanti y Wong, 1984 [57]).

El problema general de diseño de red trata de encontrar la configuración que minimiza la suma de costos fijos de aristas incluidas en el diseño y costos de transportar los bienes que son requeridos desde un punto origen a otro punto destino a través de la red creada.

La importancia de los problemas de diseño de red radica en el hecho de que son muy utilizados dadas sus innumerables aplicaciones. Desde un punto de vista teórico, hasta la versión más simple de un problema de diseño de red no capacitada con costos de ruteo lineales modela muchos de los problemas bien conocidos en optimización combinatoria (e.g., ruta más corta, mínimo árbol de expansión, agente viajero, etc.).

En los problemas específicos de diseños o rediseños de redes de telecomunicación o de computadoras, de sistemas eléctricos de potencia, de redes de transporte y otros, se pueden considerar o no capacidades finitas en los arcos o nodos de la red. Incluir capacidades en los modelos vuelve el problema más realista, pero también mucho más complejo y por lo tanto más difícil de resolver que su versión no capacitada.

En el presente trabajo se aborda un problema de diseño de redes donde varios productos (una nomenclatura general) deben ser llevados a través de una red desde sus orígenes hasta sus destinos respectivos y todos los productos viajan a lo largo de caminos o aristas con capacidades limitadas.

A cada arista potencial se le asignará un costo fijo el cual representa su costo de construcción o de utilización y un costo variable por unidad de flujo que circule por ella, el cual es dependiente del producto que será transportado y de la dirección que tomará en dicha arista. Asimismo cada arista tiene una capacidad finita que será compartida entre todos los productos que la usen sin importar la dirección del flujo.

Una aplicación específica de este problema en particular la podemos encontrar en los sistemas de telecomunicación, en donde podemos diseñar una red que conecte ciertos nodos entre sí (nodos origen con nodos destino) para garantizar el flujo de cierto o ciertos tipos de productos (información). Las conexiones que existen entre cada nodo poseen capacidades finitas para tránsito de información, además de ser una capacidad compartida para todos los productos y ambos sentidos de flujo.

Existen trabajos que han tratado los problemas de diseño de red en versiones deterministas y considerando capacidades infinitas en los arcos, como el problema de ubicación de instalaciones, en donde el diseño consiste en seleccionar un subconjunto de nodos de la red en los cuales se ubicarán plantas y en asignar clientes a esas instalaciones con el objetivo de minimizar los costos totales (Daskin, 1995 [15]). La solución de este problema incluye el número y la ubicación de instalaciones que deberán abrirse, así como su capacidad y los clientes que atenderán.

Otro tipo de problema de diseño de red determinista y de capacidad infinita para los arcos es el de seleccionar, de un conjunto de arcos potenciales, aquellos que permitan transportar un producto desde su nodo origen hasta su nodo destino a un costo total mínimo, costo que incluye los costos de transporte y los costos por utilización o creación de cada arco. Para este tipo de problemas, la demanda completa puede ser transportada por una única ruta que conecte al nodo origen con el destino de cada producto, dada la característica de capacidad infinita de los arcos.

Una variante del problema mencionado es el que incluye, para cada producto, un nodo origen y varios nodos destino en una red dirigida y no capacitada (Cruz, MacGregor y Mateus, 1998 [13]).

Una consideración extra, la de incluir capacidad en los arcos puede hacerse, con el objetivo de modelar un problema más realista, pero también más complicado de resolver.

Existen trabajos que han abordado el problema de diseño de redes con capacidad finita multiproducto con un solo nodo origen y destino por producto y en donde varios productos deben ser llevados a través de una red desde sus orígenes hasta sus destinos. Este tráfico viaja a lo largo de aristas con capacidades finitas y a cada arista potencial se le asigna un costo fijo que representa su costo de construcción o de utilización y un costo variable por unidad de flujo dependiente del producto que será transportado y de la dirección que tomará en dicha arista (Gendron y Crainic, 1996 [30]; Crainic, Frangioni y Gendron 2001 [11]; Crainic, 2000 [10]; Álvarez y De Alba, 2004 [16]).

El problema anterior es una versión determinista del problema que abordaremos en el presente trabajo en donde consideramos incertidumbre en parámetros clave, como lo son los costos variables o de transportación y las demandas para cada producto.

Existen diversas maneras de abordar un problema que contenga incertidumbre, cada manera tiene ventajas con respecto a las demás, pero también desventajas, por lo que dependiendo de las características propias del problema en cuestión, será la opción seleccionada. Las formas de abordar la incertidumbre son:

- Convertir el problema con incertidumbre a uno determinista y utilizar las técnicas de optimización determinista.
- Programación Estocástica.
- Programación Robusta.

Como veremos más adelante, convertir un problema con incertidumbre a uno determinista puede ser una función sencilla de realizar, pero el precio que se paga es muy alto, ya que la solución que encontremos al problema simplificado puede ser muy mala para nuestro problema original. Este enfoque simplemente ignora la incertidumbre.

Por otro lado, la programación estocástica nos dará buenos resultados siempre y cuando el problema que deseamos atacar sea de características repetitivas, es decir, que

sea la toma de una decisión que se repite cada cierto período. Esta característica es necesaria para la programación estocástica, que nos garantiza muy buenos resultados al largo plazo repetitivo.

Y por último tenemos la programación robusta, que nos deja trabajar con incertidumbre para problemas que serán resueltos una sola vez pero que, sin importar lo que suceda con los parámetros sujetos a variación, deseamos tener un buen desempeño de la decisión que tomemos.

Un trabajo en el que se incluye incertidumbre y se aplica la programaciòn robusta es en el del problema robusto de abastecimiento capacitado internacional, el cual consiste en seleccionar un subconjunto de proveedores de un conjunto disponible de los mismos, localizados internacionalmente. Los proveedores seleccionados deben de satisfacer la demanda de artículos para un conjunto de plantas que se encuentran a lo largo del mundo (Laguna y González-Velarde, 2004 [52]).

Adicionalmente, entre las formas de tratar con incertidumbre se encuentra un criterio que evalúa el desempeño de la solución del peor caso, este criterio es el minimax, el cual puede ser aplicable cuando un costo muy alto en cualquier escenario es extremadamente indeseable. Esta técnica es importante también para los administradores de decisión adversos al riego, interesados en el desempeño de la distribución de la solución.

Los criterios minimax son de naturaleza conservadora, basados en la apreciación de que lo peor puede suceder. Se aplica este criterio cuando es importante para el administrador que sus decisiones, aún en el peor de los casos, resulte en beneficios.

Entre las obras que han atacado problemas de diseño de red con incertidumbre se cuentan la del problema de diseño de red con criterio minimax, en donde es necesario definir la red robusta (para todos los escenarios definidos) que tenga la mejor realización para el “peor caso” sobre todos los escenarios posibles y que pueda transportar un cierto material demandado desde un punto origen a uno destino (Herrmann, 1999 [37]).

Por otra parte, la literatura también ha descrito el problema de diseño de red con incertidumbre, similar al abordado en este trabajo, pero con capacidades infinitas en las aristas. Para resolverlo se hace uso de un truco de modelación y es gracias a esta

implementación que pueden resolver el problema mediante una adaptación de la metodología de Benders (Kouvelis y Yu, 1997 [50]).

Nuestro problema es más complicado y no acepta trucos de separación o consideraciones que lo transformen a una versión simplificada, por el hecho de que usaremos aristas con capacidades finitas, y hacer una consideración especial, sería ignorar los límites de cada arista.

Ya hemos mencionado que la consideración de incertidumbre, en un problema de diseño de red en general, complica sobremanera la resolución de dicho problema y podríamos preguntar si son necesarias realmente estas complicaciones. La respuesta es sí, ya que esta consideración de variantes en los datos que tomamos como entrada nos permite ver de manera más amplia nuestro verdadero problema, que emula más cerca la realidad. Esta visión ampliada de un problema y sus consideraciones son de capital importancia en problemas de planeación a largo plazo.

Otros trabajos que han tratado problemas de diseño de red con incertidumbre en parámetros son los abordados por Averbakh y Berman (2000 [2]); Gutiérrez y Kouvelis (1995 [34]); Gutiérrez, Kouvelis y Kuruwarwala (1995 [35]); Haight, Ralls y Starfield (2000 [36]); Kennington, Lewis, Olinick, Ortynsky y Spiride (2003 [46]); Killmer, Anandalingam y Malcolm (2001 [48]); Raja y Han (2005 [64]); Snyder y Daskin (2003 [67]); Yu y Li (2000 [70]).

Considerar incertidumbre, en la demanda por ejemplo, nos permite construir un diseño de red que no solo será bueno durante las épocas de demanda más común o esperada (como podría suceder con el diseño arrojado por la solución de un problema determinista, con datos promedio o esperados), sino que será bueno también para cuando la demanda no alcance los niveles promedio o cuando incluso los sobrepase.

Por otro lado, considerar incertidumbre en los costos variables nos facilitará desarrollar un diseño que procure un buen desempeño en cuanto a costos de transportación, sin importar que los mismos fluctúen por estaciones o factores externos.

En resumen, el considerar incertidumbre en nuestro problema nos dará la capacidad de planear para períodos a largo plazo, en vez de tener que conformarnos con soluciones buenas durante un corto tiempo o durante un breve lapso de un ciclo.

1.2 Objetivos de la Tesis.

Se ha mencionado que el problema general de diseño de redes ha sido objeto de diversos estudios debido principalmente a sus múltiples aplicaciones. En este trabajo de diseño de red con incertidumbre se aborda y se soluciona el siguiente problema de toma de decisiones:

Para un problema de diseño de red multiproducto con capacidades finitas en las aristas e incertidumbre en parámetros de entrada, ¿cuál es el diseño que ofrece mejor desempeño con respecto a todos los escenarios posibles de dichos parámetros? Entenderemos desempeño como el costo del diseño propuesto comparado con el mejor costo posible en cada escenario.

El costo de un diseño de red en un escenario específico esta determinado por la suma de los costos fijos de las aristas seleccionadas para pertenecer a la red y el costo por ruteo de bienes a través de la red definida por dichas aristas capacitadas, considerando los datos de dicho escenario.

Considerando lo anteriormente expuesto, los objetivos de esta tesis son los siguientes:

- Realizar un estudio de la estructura del problema de diseño de red multiproducto con capacidades en las aristas sujeto a incertidumbre en demandas y costos variables o de transporte.
- Investigar y derivar técnicas de solución para resolver el problema de diseño de red con capacidad finita e incertidumbre bajo estudio.
- Desarrollar e implementar un sistema de optimización basado en técnicas inteligentes de búsqueda para resolver el problema de diseño de red capacitada multiproducto con incertidumbre bajo estudio.

1.3 Alcance del Trabajo.

Como resultado del presente trabajo se entrega un sistema computacional que aplica los procedimientos propuestos al problema de diseño de redes multiproducto con capacidad finita, cargos fijos e incertidumbre en demanda y costos variables.

El total de programas desarrollados que integran dicho sistema produce un conjunto de soluciones, esto es, diseños de red de calidad en un formato extendido que especifica los costos fijos y costos variables incurridos, así como las rutas de envío de flujo y la cantidad de este flujo para cada uno de los productos que integran el problema y, para cada uno de los posibles escenarios de datos.

Los programas fueron desarrollados usando el lenguaje de programación C y C++ desarrollados en Visual C++, optimizados en CPLEX ([9]) e implementados en una estación de trabajo Sun Ultra 10 con sistema operativo Solaris versión 5.9 y en una computadora personal con procesador Pentium 4 a 2.8 Ghz. con sistema operativo Windows XP Profesional y 512 MB de memoria RAM. Para la ejecución de los programas desarrollados se requiere un formato específico de instancias de red, mismo que se explicará detalladamente en el capítulo relativo a los resultados computacionales.

El sistema desarrollado fue probado con redes de hasta 50 nodos, 50 productos y 920 aristas, entregando soluciones de buena calidad en tiempos razonables.

1.4 Justificación.

Como ha sido mencionado, en la literatura especializada aparecen varios trabajos sobre la versión no capacitada del diseño de una red multiproducto (Hoang, 1973 [39]; Dionne y Florian, 1979 [20]; Barr, Glover, y Klingman, 1981 [4]; Magnanti y Wong, 1984 [57]; Balakrishnan y Magnanti, 1989 [3]; Holmberg y Migdalas, 1991 [42]). Para el caso que considera capacidades finitas en las conexiones encontramos algunos trabajos en redes uniproducto (Khang y Fujiwara, 1991 [47]; Erickson, Monma y Veinott, 1987 [22]), así como algunos otros trabajos para redes multiproducto (Gendron y Crainic, 1996 [30]; Crainic, Frangioni y Gendron 2001 [11]; Crainic, 2000 [10]). La

mayoría de los trabajos mencionados, consideran redes orientadas, asociando una capacidad a cada arco de la red, y en ninguno de ellos se maneja incertidumbre.

Otras publicaciones referentes a problemas de diseño de red son las escritas por Crainic, Gendreau y Farvolden (2000 [12]); Fisher y Jaikumer (1978 [24]); Florian, Guerin y Bushel (1976 [26]; Gabow, Goemans y Williamson (1993 [27]); Geoffrion y Graves (1974 [31]); Holmberg y Yuan (1998 [43]); Holmberg y Yuan (2000 [44]); Magnanti, Mirchandani y Vachani (1995 [55]); Richardson (1976 [65]).

Entre todos estos trabajos publicados, ninguno puede resolver siquiera la versión determinista del problema que analizamos, debido principalmente a las capacidades asociadas a las aristas de las que hacemos uso. Una arista es una unión no orientada entre 2 nodos, y asociada a ella existen 2 arcos orientados que son la unión dirigida de los mismos nodos; esto significa que la arista $\{i, j\}$ está formada por los arcos orientados (i,j) y (j,i). Tradicionalmente cuando se han considerado capacidades en problemas de diseño de red, los autores las consideran en los arcos y no en las aristas, como nos incumbe. En nuestro caso, la capacidad asignada a la arista es compartida por los dos arcos.

Existe un par de trabajos que consideran aristas con capacidades para el problema que nos interesa (De Alba, 2004 [16]; Álvarez y Cobos, 2004 [8]), pero aunque el problema es en esencia el mismo que abordamos en esta obra, ambos solo modelan la versión determinista del problema.

En resumen, sólo existe un trabajo previo al nuestro que ha manejado incertidumbre para el problema de diseño de red multiproducto con aristas capacitadas (Medina, 2005 [61]) en el que se desarrolló una heurística con criterio de optimización minimax. Los resultados de este trabajo serán comparados con los resultados de la presente obra.

Publicaciones interesantes que abordan problemas con incertidumbre bajo diversos enfoques son los publicados por Darlington, Pantelides, Rustem y Tanyi (2000 [14]); Eppen, Kipp y Schrage (1989 [21]); Kouvelis, Kurawarwala y Gutiérrez (1992 [49]); Laguna, Lino, Pérez, Quintanilla y Valls (2000 [53]); Mausser y Laguna (1998 [58]); Mausser y Laguna (1999 [59]); Rosenhead, Elton y Gupta (1972 [66]); Trafalis, Mishina y Foote (1999 [69]).

En el desarrollo del presente trabajo, no se emplea un criterio de optimización minimax, sino que se manipula el riesgo igualándolo con la variabilidad de las salidas desconocidas. Específicamente se propone una función objetivo que combina los dos primeros momentos con el propósito de minimizar el costo esperado y el riesgo.

Si bien la clase de problemas que nos ocupa pueden formularse como un programa no-lineal entero mixto, el tamaño de instancias reales impide la obtención de soluciones mediante los paquetes comerciales de optimización disponibles. La mayoría de los métodos de solución propuestos para problemas de diseño de red óptima en general, se basan en el hecho de que este problema es NP-completo (Garey y Jonson, 1979 [28]; Hochbaum y Segev, 1989 [40]).

La interacción entre costos fijos y variables al construir una solución (esto es, un diseño), así como la relación entre la capacidad finita de las aristas (compartida por cada producto que trata de circular en la red) y los costos fijos de esas aristas, elevan enormemente la dificultad al resolver este problema cuando nos enfrentamos a instancias realistamente grandes (Johnson, Lenstra y Rinnooy, 1978 [45]; Gendron, 2002 [29]). Aunado a lo anterior, la consideración de incertidumbre en algunos de los parámetros de este problema lo convierte en un problema aún más complejo y, por lo tanto, más difícil de resolver.

Hasta el momento no se conocen algoritmos exactos que puedan resolver tales problemas en un período razonable de tiempo, por lo que se ha hecho necesario el desarrollo de técnicas aproximadas o heurísticas que aporten soluciones, que si bien no hay garantía de que constituyan el óptimo global, poseen una alta calidad y se encuentran de forma relativamente sencilla y rápida.

Por ello se ha considerado novedoso y de gran importancia la realización del presente trabajo, dada la necesidad de desarrollar métodos específicos de solución que encuentren soluciones de alta calidad al problema de diseño de red multiproducto con capacidades en las aristas e incertidumbre en parámetros, el cual ha sido muy poco tratado en la literatura.

1.5 Estructura de la Tesis.

Esta tesis se presenta con una introducción al problema que se aborda (capítulo 1); describiéndolo y definiendo su relevancia en diversas áreas de la Investigación de Operaciones, así como antecedentes del problema de diseño de red de aristas capacitadas multiproducto con incertidumbre (capítulo 2).

Posteriormente se plantea el problema y se presenta la formulación matemática usada para definir y resolver el problema tratado (capítulo 3), para mostrar seguido a esto, la metodología de solución, que incluye una breve descripción de las técnicas empleadas (capítulo 4), la forma en que se diseñaron los experimentos y el ajuste de parámetros (capítulo 5) así como la evaluación computacional (capítulo 6).

Las conclusiones y recomendaciones se incluyen para futuros trabajos (capítulo 7). Se añaden a esta tesis algunas tablas y anexos para la mejor comprensión y comparación de los resultados obtenidos en el apartado de Anexos.

CAPÍTULO 2. MARCO TEÓRICO

2.1 Introducción.

Una amplia compilación de modelos para el problema general de diseño de redes puede ser encontrada en Magnanti y Wong (1984 [57]). La versión orientada y determinista de este problema con capacidad infinita en las conexiones ha sido estudiada ampliamente (Magnanti, Mireault y Wong, 1986 [56]; Balakrishnan y Magnanti, 1989 [3]; Holmberg y Hellstrand, 1998 [41]), así como el caso de la versión orientada con capacidad finita (Gendron y Crainic, 1996 [30]; Crainic, Frangioni y Gendron 1999 [11]; Crainic, 2000 [10]).

El problema determinista en redes no orientadas con capacidad finita para las conexiones o aristas, ha sido menos estudiado, se conocen solo un par de investigaciones (De Alba, Álvarez y González-Velarde 2003 [17]; Cobos, 2004 [7]), en donde se reportan buenos resultados empleando metaheurísticas como Búsqueda Dispersa y Búsqueda Tabú respectivamente, así como una técnica ascendente dual para encontrar buenas cotas inferiores y soluciones cercanas a la óptima (Herrmann, Ioannou, Minis y Proth, 1996 [38]).

Considerando incertidumbre en parámetros de entrada, existen también trabajos sobre la versión orientada de capacidad infinita (Kouvelis y Yu, 1997 [50]), pero en específico para el problema no orientado de capacidad finita en las aristas y parámetros bajo incertidumbre solo se conoce en la literatura una obra hasta la fecha (Medina, Álvarez y De Alba, 2004 [60]).

En el trabajo de Kouvelis, como consideran capacidades infinitas en los arcos, y pueden utilizar un truco de modelación, en donde suponen que debe ser transportada tan solo una unidad de flujo desde el origen hasta el destino de cada producto, de tal suerte que pueden usar el costo de transportación de un nodo *i* a uno *j* para representar el uso

del arco *(i,j)* para transportar todo el flujo de cierto producto desde el nodo *i* al *j* (Kouvelis y Yu, 1997 [50]). Gracias a este artificio, un diseño factible para un escenario es factible para cualquier otro, sin requerir ninguna comprobación adicional y de esta forma pueden resolver el problema mediante una adaptación de la metodología de Benders para la generación de un diseño robusto de red, utilizando un criterio de desempeño minimax. Sin embargo, las características de capacidad finita en nuestro problema, hace que sea imposible el uso de un artificio como el mencionado anteriormente.

En este punto, vale la pena aclarar por qué no es posible transformar la red no dirigida en una dirigida dentro del problema bajo estudio y utilizar entonces técnicas desarrolladas para ellas: en una red no dirigida si existen 2 nodos *i* y *j*, entonces existirán siempre ambos arcos *(i, j)* y *(j, i)*, mientras que en las redes dirigidas pudiera solo existir uno de esos dos arcos conectando a los nodos *i* y *j*. Por otra parte, las redes dirigidas consideran que cada arco posee su propia capacidad, esto es, la capacidad de un arco es independiente de la capacidad de cualquier otro arco, mientras que en las redes no dirigidas la capacidad de la arista es compartida por los dos arcos que la componen.

Esto significa que en redes no orientadas un arco puede usar la capacidad total correspondiente a su arista si y solo si el arco opuesto no está siendo usado o, podrá usar una capacidad residual si su correspondiente arco de sentido opuesto ya ha usado parte de ella o, un arco no tendrá capacidad disponible para uso si su correspondiente arco de sentido opuesto ya ha usado la capacidad total de la arista.

2.2 Enfoques Tradicionales para Manejar Incertidumbre.

Dado que la incertidumbre afecta a un amplio rango de administradores, ingenieros, y otros tomadores de decisiones, la mejor manera de manejarla y de tomar decisiones bajo ella es aceptarla, estructurarla y entenderla, para finalmente, hacerla parte del razonamiento de toma de decisión.

Las situaciones de decisión se han dividido convencionalmente (Kouvelis y Yu, 1997 [50]) en tres categorías:

a. Con certidumbre

b. Bajo riesgo y
c. Con incertidumbre

En las situaciones de certidumbre con objetivo único (en donde no hay elementos de azar que intervengan entre la decisión y el resultado de la misma), con un único objetivo, aplicamos optimización determinística para seleccionar la decisión con el valor más favorable en su función objetivo.

En las situaciones de riesgo, (en donde los lazos entre decisiones y salidas son probabilísticos), usualmente usamos optimización estocástica para optimizar el valor esperado de un solo objetivo.

Las situaciones de incertidumbre ocurren cuando es imposible atribuir probabilidades a los resultados de ninguna decisión. Esto puede ocurrir por ejemplo, cuando el resultado de una decisión depende de una decisión simultánea o subsiguiente de un competidor cuyos objetivos entran en conflicto con los nuestros, o de un futuro evento externo de variedad no repetitiva para el cual las estimaciones de probabilidades son un ejercicio cuestionable.

Es posible convertir un problema con incertidumbre en un problema de riesgo, por ejemplo, con la estimación subjetiva de probabilidades, y entonces usar un modelo de optimización estocástica para resolverlo. Es también posible convertirlo en un problema de certidumbre, por ejemplo, con el uso de un escenario futuro esperado, y entonces resolver el problema resultante con optimización determinística.

La optimización determinística es alimentada con los datos de una instancia del modelo de decisión. Este enfoque ignora completamente la incertidumbre o bien, trata de usar la información histórica para pronosticar el futuro. La instancia seleccionada de la información de entrada representa el estimador más probable a suceder.

La mayor debilidad del enfoque determinístico es su inhabilidad de reconocer la presencia de otras instancias de datos distintas de la “más probable” la cual se usa para generar la decisión “óptima”. Para estas instancias, las decisiones generadas pueden ser substancialmente suboptimales. Esto es inaceptable para quienes toman decisiones, y que viven con las consecuencias de la decisión tomada a pesar del escenario realizado en el futuro, aún y cuando la decisión haya sido “óptima” para el escenario futuro “más probable”.

Cualquier enfoque que falle en reconocer otras instancias y no haga ajustes en las decisiones generadas para los escenarios futuros distintos del “más probable” son inaceptables para ambientes de decisión en presencia de incertidumbre significante.

El enfoque de la optimización estocástica reconoce la presencia de múltiples instancias de datos que pueden ser realizadas potencialmente en el futuro. Típicamente el modelo de decisión intenta generar una decisión que maximice (o minimice) una medida de rendimiento esperado, donde la expectativa es tomada en cuenta sobre la distribución de probabilidad.

La interpretación tradicional de una decisión “estocásticamente óptima” es la de optimalidad a largo plazo es decir, si se debe tomar una decisión una y otra vez y las instancias de información llegan de manera aleatoria según la distribución de probabilidad asumida, en el largo plazo repitiendo esta decisión uno logra el máximo rendimiento.

Este enfoque forza al tomador de decisiones a asignar probabilidades a las varias instancias de datos para los escenarios futuros. Asignar probabilidades o designar distribuciones de probabilidad es mucho más que un ejercicio trivial para muchos tomadores de decisiones, particularmente cuando los ambientes de decisión tienen múltiples factores interdependientes con incertidumbre.

La falla más importante de la optimización estocástica y determinística, es que no pueden reconocer que asociado con cada decisión hay toda una distribución de resultados, dependiendo de cuál escenario de datos se realizó realmente, y cualquier enfoque que evalúe decisiones usando solo un escenario de datos, ya sea el esperado o el más probable de suceder, está destinado a fallar.

Un administrador adverso al riesgo estará razonablemente más interesado en el peligro que existe de tener un pobre desempeño del sistema para algunos de los escenarios, que en optimizar un desempeño esperado en el sistema sobre todos los escenarios potenciales o simplemente en el desempeño del escenario más probable.

Esto es particularmente importante para decisiones de naturaleza única y por lo tanto realizadas solamente una vez. Optimalidad a largo plazo para estas situaciones es irrelevante, dado que el elemento de repetición de la decisión se encuentra ausente, y el

escenario “mas probable” o el esperado son solamente un subconjunto de los escenarios potencialmente realizables.

Quien toma las decisiones sabe que, para cualquier escenario potencialmente realizable, él debe vivir con las consecuencias, para el desempeño del sistema, de la decisión tomada. Por lo tanto, el desempeño de una decisión a través de todos los escenarios realizables es muy importante.

Lo que el administrador o tomador de decisiones quiere no es el “óptimo” para un escenario específico (aún y cuando éste sea el escenario “más probable”) ni el “óptimo a largo plazo”, sino una decisión que tenga un buen desempeño a través de todos los escenarios. En nuestra terminología, lo que el administrador quiere dentro de ambientes de decisión con incertidumbre significante es una decisión robusta, una que se desempeñe bien sobre todos los escenarios, incluso sobre el peor de todos. El objetivo de este enfoque es producir decisiones que tendrán un valor objetivo razonable bajo cualquier escenario de nuestro modelo de decisión sobre un horizonte de planeación.

Una solución al programa matemático es robusta con respecto a optimalidad (solución Robusta), si se mantiene cerca de optimalidad para cualquier escenario de entrada del modelo. Una solución es robusta con respecto a factibilidad (modelo Robusto) si se mantiene casi siempre factible para la realización de cualquier escenario (Mulvey, Vanderbei, Zenios, 1995 [62]).

Más propiamente, al referirnos al término robusto lo que pretendemos es desarrollar un diseño de red que sea en algún sentido “bueno” para cualquier futuro concebible. Esta solución no necesariamente es óptima para alguno de los posibles futuros en particular, y de hecho frecuentemente no lo será.

Para los trabajos que consideran incertidumbre en datos clave del problema se han aplicado diversos métodos de solución, como algoritmos genéticos para un problema minimax de diseño de red con demanda y probabilidad de cada escenario inciertas (Herrmann, 1999 [37]). En esta obra se mantienen dos poblaciones que evolucionan simultáneamente, una de soluciones y la otra de escenarios. La convergencia de ambas poblaciones es hacia la solución robusta y hacia el escenario del peor caso, respectivamente. La red con la que se trabaja aquí es dirigida y no capacitada.

El modelo robusto introducido por Mulvey, Vanderbei, y Zenios (1995 [62]), incluye una función objetivo con los momentos máximos de la distribución en una función de costo, considerando un término de penalidad por factibilidad. Esta metodología ha sido utilizada exitosamente en diversas aplicaciones como en el problema de ubicación internacional de recursos (Laguna y González-Velarde, 2004 [52]), o en el problema de expansión de capacidad en redes de telecomunicaciones (Laguna, 1997 [51]).

La técnica aplicada en el problema de diseño de una red de distribución internacional utiliza una heurística basada en la descomposición de Benders y búsqueda Tabú (Laguna y González-Velarde, 2004 [52]). En esta red las capacidades se encuentran en los nodos origen o plantas proveedoras, los arcos son dirigidos y se considera incertidumbre en la demanda y los tipos de cambios. En este trabajo, se aplica una función objetivo que posee una medida del riesgo.

Como el problema que nos incumbe está sujeto a variación en algunos de sus parámetros, y la simple solución al problema determinístico no es suficiente, debemos aplicar técnicas de modelación que puedan trabajar con incertidumbre en los datos de entrada, como lo puede ser la optimización estocástica o bien, como se decidió hacer, la optimización robusta.

La decisión de emplear optimización robusta se hace porque, como se ha demostrado en reciente literatura de investigación, muchos problemas combinatorios difíciles, como la asignación cuadrática y los problemas de diseño de red multiproducto, exhiben la propiedad importante de tener múltiples, y significantemente distintas soluciones con pocos puntos porcentuales de distancia del óptimo. En estos problemas, éste es el rasgo principal que los hace adecuados para desarrollar algoritmos que puedan encontrar una solución común cerca del óptimo (robusta) para una variedad de escenarios operativos futuros.

Por último, es interesante destacar que la necesidad de considerar la presencia de incertidumbre en aplicaciones reales surge según el enfoque que deseemos. Si nuestro interés no se encuentra enfocado en los parámetros que presentan incertidumbre, o son de importancia secundaria podemos considerar problemas deterministas con un valor esperado, o fijando su valor a la cantidad, que pueda garantizar un nivel de servicio

requerido; pero si la variable que fluctúa es de mucha importancia en nuestro modelo, o se trata de la razón de ser de nuestro problema, el considerar un valor esperado o un nivel fijo para este valor puede entregarnos soluciones sustancialmente subóptimas.

2.3 Técnicas Usadas para Resolver Problemas de Diseño.

Diferentes técnicas, tanto heurísticas como metaheurísticas, han sido aplicadas frecuentemente en la resolución de problemas de diseño de red. Entre ellas, GRASP ha tenido un especial buen desempeño, por lo que se seleccionó para guiar la construcción de soluciones al problema bajo estudio.

2.3.1 Heurísticas y Metaheurísticas.

Existe un tipo de algoritmos llamados *heurísticas* (del griego heuriskein, que significa “encontrar”) cuya finalidad es encontrar soluciones “satisfactorias” en un tiempo “razonable”. El escaso rigor matemático con el que fueron originalmente calificadas (ambos términos “satisfactorio” y “razonable” pertenecen al orden de lo subjetivo), fue superado por sus implicaciones prácticas y como herramientas útiles para dar solución a problemas tradicionalmente complejos.

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 situaciones, 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 situaciones.

Los problemas de decisión que normalmente se presentan en la comunidad empresarial, y donde por lo general existen recursos escasos o bien, requerimientos mínimos a cumplir, ejemplifican situaciones donde es necesario encontrar la respuesta óptima al problema planteado. A este tipo de problemas se les llama problemas de optimización.

Existe un tipo particular de problemas de optimización que son de gran interés para los métodos heurísticos: los llamados problemas de optimización combinatoria. En estos problemas, las variables son enteras, y por lo general el espacio de soluciones está formado por ordenamientos o subconjuntos de números naturales (de ahí su nombre) (Díaz et al., 1996 [19]). En este tipo de problemas, cuya complejidad crece exponencialmente con el tamaño del problema, los métodos heurísticos se acercan, o se pretende que se acerquen a la solución óptima en un tiempo razonable.

Una definición sencilla de los métodos heurísticos es la siguiente: "procedimientos simples, a menudo basados en el sentido común, que se supone ofrecen una buena solución (aunque no necesariamente la óptima) a problemas difíciles de un modo fácil y rápido" (Zanakis y Evans, 1981 [71]).

La aplicación de heurísticas es recomendable en diferentes circunstancias, por lo general en los casos en los que:

- No existe un método exacto de solución
- No se requiere la solución óptima
- Los datos son poco fiables, o se hace uso de un modelo demasiado simplificado de la realidad
- Existen limitaciones de tiempo y espacio

También se usan frecuentemente como paso intermedio para otras aplicaciones.

Existen varios tipos de heurísticas, según el modo en que buscan y construyen soluciones. Una posible clasificación es la siguiente:

a. Basadas en métodos constructivos. Aquéllas que paulatinamente añaden elementos individuales a la solución que van formando hasta que obtienen una solución factible. El más popular de estos métodos es el método voraz el cual construye paso a paso la solución buscando el máximo beneficio en cada paso.
b. Basadas en métodos de descomposición. Aquéllas que subdividen un problema en subproblemas más pequeños, usando la salida de uno como la entrada de otro, de forma que al resolverlos todos tengamos una solución para el problema global.
c. Basadas en métodos de reducción. Tratan de identificar alguna característica que deba poseer la solución óptima y de ese modo simplificar el problema. Como

ejemplo de este caso se trataría de identificar valores que deban tener ciertas variables, o identificar variables correlacionadas, etc.

d. Basadas en una manipulación del modelo. Modifican la estructura del modelo a fin de hacerlo más sencillo de resolver, deduciendo, a partir de su solución, la solución del problema original.
e. Basadas en métodos de búsqueda por entornos. Dentro de esta última clasificación se encuadran la mayoría de las heurísticas que se aplican actualmente. Estos métodos parten de una solución factible inicial, y mediante alteraciones, van pasando a otras factibles de su entorno, almacenando la mejor de las soluciones encontradas hasta el momento.

Generalmente, los métodos heurísticos terminan con un óptimo local al problema que están manipulando, lo cual no es deseable la mayoría de las veces, sobre todo, en aquellos problemas con múltiples óptimos locales de baja calidad. Una metaheurística es una estrategia maestra que guía y modifica otras heurísticas para producir soluciones más allá de aquéllas normalmente generadas en una búsqueda para optimalidad local.

Las metaheurísticas en su definición original son métodos de solución que organizan una interacción entre procedimientos de mejora local y estrategias de más alto nivel para crear un proceso capaz de escapar de óptimos locales y ejecutar una búsqueda más completa del espacio de soluciones (Glover, Laguna y Martí, 2003 [33]). Las metaheurísticas se clasifican a su vez en metaheurísticas con memoria y sin memoria.

A diferencia de las metaheurísticas sin memoria, las metaheurísticas con memoria poseen estructuras que retienen información sobre decisiones tomadas con anterioridad permitiendo cierta forma de aprendizaje. Dentro de las metaheurísticas con memoria tenemos la Búsqueda Tabú (Glover y Laguna, 1997 [32]), y la Búsqueda Dispersa, mientras que en las metaheurísticas sin memoria, cuando se consideran en su descripción original, tenemos el Recocido Simulado, los Algoritmos Genéticos, y el GRASP. No obstante, vale la pena señalar que la implementación de estos métodos en la actualidad cada vez muestra más aspectos comunes, por lo que algunos autores ya comienzan a utilizar el nombre de programación de memoria adaptativa (Taillard et al., 2001 [68]).

2.3.2 GRASP.

GRASP es una de las metaheurísticas más jóvenes, surgidas durante la década de los 80 con el fin de resolver problemas difíciles en el campo de la optimización combinatoria, específicamente GRASP fue desarrollado por Feo y Resende (1989 [23]) para estudiar un problema de cobertura de alta complejidad.

La palabra GRASP es el acrónimo de *greedy randomized adaptative search procedure* (procedimiento de búsqueda miope, aleatorizado y adaptativo). Cada una de estas palabras caracteriza una de las componentes de esta metaheurística, la cual a diferencia de sus parientes mayores que operan sobre soluciones obtenidas previamente con el fin de mejorarlas, dirige la mayor parte de su esfuerzo a construir soluciones de alta calidad que son posteriormente procesadas con el fin de conseguir otras aún mejores. En otras palabras, GRASP es una metaheurística para generar soluciones aproximadas (i.e. subóptimas de buena calidad, pero no necesariamente óptimas) a problemas de optimización combinatoria.

GRASP es un método multiarranque, en el cual cada iteración consiste en la construcción de una solución miope aleatorizada seguida de un post-procesamiento que intenta mejorar la solución construida. Existen dos pasos en el procedimiento: la fase de construcción de una solución y la fase de mejoramiento.

En la mayoría de las adaptaciones del GRASP la fase de construcción tiene como características principales las medidas adaptativas miopes y la selección aleatoria. Posteriormente se han incluido diseños de memoria adaptativa (Fleurent y Glover, 1999 [25]) para retener y analizar las características de las soluciones seleccionadas y así obtener una base para mejorar la ejecución posterior del proceso constructivo.

Los principales componentes de GRASP para la fase de construcción son una función de evaluación miope, un procedimiento de elección al azar y un proceso de actualización adaptativo. Una función miope o enfoque de tipo miope, consiste en efectuar el mejor movimiento (o tomar la mejor decisión) disponible en cada paso y el término adaptativo se refiere al hecho de que la función considera decisiones tomadas en pasos anteriores, previos al momento de considerar la siguiente decisión.

Todos los mecanismos de construcción considerados construyen una solución incorporando un elemento a la vez, así en cada paso del proceso de construcción, se tiene a la mano una solución parcial.

Un elemento que puede seleccionarse como parte de una solución parcial se llama elemento candidato. Para determinar qué elemento candidato debemos seleccionar para incluirse en una solución parcial, generalmente se hace uso de una función miope, la cual mide la contribución local de cada elemento candidato a la solución parcial. La elección miope sería agregar el elemento candidato con el mejor valor de dicha función.

Para introducir aleatoriedad a este procedimiento se hace uso de una lista restringida de candidatos (LRC). Esta lista, empleada en la fase de construcción, está formada por elementos de alta calidad, es decir, los elementos candidatos con los mejores valores de la función miope, de los cuales se escogerá uno aleatoriamente. Esta lista puede consistir de un número fijo de elementos (restricción por cardinalidad) o elementos con los valores de la función miope dentro de cierto rango. Una vez agregado un elemento candidato a la solución parcialmente construida, deben recalcularse los valores de la función miope. Esto hace que el procedimiento adquiera la característica de adaptabilidad que le es particular.

En la fase de construcción se intenta obtener una solución tentativa, la cual es posteriormente mejorada. En las primeras aplicaciones de GRASP, la fase de post-procesamiento consistía únicamente en una búsqueda local que tenía como punto de partida la solución obtenida en la fase constructiva. Actualmente se usan también otras heurísticas y metaheurísticas para post-procesar las soluciones, por ejemplo Búsqueda Dispersa, etc.

La fase de post-procesamiento es necesaria dado que el proceso de construcción de GRASP no garantiza optimalidad local, incluso con respecto a entornos definidos de forma muy simple. La idea que apoya la funcionalidad de GRASP es que este método produce una diversidad de buenas soluciones, esperando que al menos una de ellas esté en un entorno de la solución óptima, o al menos en la de una solución muy cercana al valor óptimo.

CAPÍTULO 3. PLANTEAMIENTO DEL PROBLEMA

3.1 Introducción.

En este capítulo se presenta el modelo matemático del problema de diseño de redes no orientadas multiproducto con capacidad finita y cargos fijos en las conexiones o aristas, así como incertidumbre en parámetros de entrada.

La incertidumbre se considerará en los siguientes parámetros:

- La demanda o volúmenes de productos a transportar, que es un parámetro clave sujeto a variación, generalmente de período a período o de ciclo a ciclo en cualquier aplicación real de diseño de red, y
- Los costos variables o de ruteo, los cuales son otro parámetro que comúnmente varía a lo largo del tiempo.

La forma en que se representará la incertidumbre en nuestra modelación será por medio de un conjunto de escenarios, que son posibles situaciones futuras, que contendrán una combinación de demandas y costos variables para cada producto, así como una probabilidad de ocurrencia asociada a dicho escenario.

3.2 Modelo Matemático.

Sea $G= (N,E)$ un grafo que representa una red no orientada con un conjunto N de nodos y un conjunto E de aristas potenciales. A cada arista $\{i,j\}$ corresponden dos arcos orientados (i,j) y (j,i). Al conjunto de arcos orientados le llamaremos A.

Para trabajar con incertidumbre, bajo el enfoque de optimización robusta debemos representar de alguna manera los valores posibles que podrán tomar los parámetros bajo incertidumbre (demanda y costos variables). Utilizamos un enfoque basado en escenarios para representar la incertidumbre en los datos de nuestro modelo

de decisión. Un escenario específico de datos representa una realización potencial (lo cual ocurre con alguna probabilidad no nula pero posiblemente desconocida) de los parámetros importantes del modelo de decisión. Denotemos por S el conjunto de escenarios potenciales.

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. Sea K el conjunto de productos con demanda d^s_k para el k-ésimo producto en el escenario $s \hat{I} S$.

Por otra parte, a cada arista potencial $\{i,j\}$ se le asigna un costo fijo f_{ij} por su utilización o construcción, así como costos c^s_{ijk} y c^s_{jik} por unidad de producto transportado el cual depende del tipo de producto de que se trate y del sentido en que circule por la arista para el escenario s. 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.

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) para cada escenario s y que denotaremos como x^s_{ijk}.

En resumen, tenemos:

Conjuntos:

N	Conjunto de nodos
E	Conjunto de aristas disponibles
A	Conjunto de arcos asociados a las aristas disponibles
K	Conjunto de productos
S	Conjunto de escenarios

Parámetros:

f_{ij}	Costo Fijo de la arista $\{i,j\}$
c^s_{ijk}	Costo unitario del producto k, en el arco (i,j) bajo el escenario s
d^s_k	Demanda del producto k en el escenario s

u_{ij} Capacidad de la arista $\{i,j\}$

p_s Probabilidad de ocurrencia del escenario s

Variables:

x^s_{ijk} Flujo del producto k en el arco (i,j) en el escenario s

$$y_{ij} = \begin{cases} 1 \text{ si la arista } \{i,j\} \text{ es incluida en el diseño} \\ 0 \text{ en caso contrario} \end{cases}$$

Es importante distinguir en el modelo que, una vez que se haya decidido conectar dos nodos i, j (esto es, la arista $\{i,j\}$ formará parte del diseño de la red), se permitirá flujo en ambos sentidos o sea, se considerarán en la red ambos arcos (i,j) y (j,i).

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

$$\sum_{\{j:(i,j)\in A\}} x^s_{ijk} - \sum_{\{j:(j,i)\in A\}} x^s_{jik} = \begin{cases} d^s_k & \text{si } i = \text{O(k)} \\ -d^k_k & \text{si } i = \text{D(k)} \\ 0 & \text{de otro modo} \end{cases} \qquad \forall\, k \in K, \forall\, i \in N, \forall\, s \in S \qquad (1)$$

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

$$\sum_{k\in K} \left(x^s_{ijk} + x^s_{jik}\right) \le u_{ij} y_{ij} \qquad \forall \{i,j\} \in E, \forall s \in S \qquad (2)$$

Estas restricciones no solo aseguran que sean respetadas las capacidades de las aristas, sino también forzan a que el flujo de cualquier producto x^s_{ijk} sea cero para todo escenario s si la arista $\{i, j\}$ no ha sido seleccionada en el diseño.

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

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

La función objetivo que proponemos tiene el propósito de minimizar el costo esperado y el riesgo. La función que considera el riesgo, penaliza solamente las desviaciones no deseadas, en contraste con los modelos de media-varianza, donde se penalizan todas las desviaciones.

$$\min \sum_{s \in S} p_s \left(\sum_{\{i,j\} \in E} f_{ij} y_{ij} \ + \ z_s \right) + w \sqrt{\frac{\sum_{s \in S^+} p_s \left(z_s - E(z_s) \right)^2}{\sum_{s \in S^+} p_s}} \tag{4}$$

En donde:

$$z_s = \sum_{k \in K} \sum_{(i,j) \in A} c_{ijk}^{s} x_{ijk}^{s}$$
$$E(z_s) = \sum_{s \in S} p_s z_s$$
$$S^+ = \{ s : z_s - E(z_s) \geq 0 \}$$

w = Ponderación para la penalización

La segunda parte de (4) es el término de aversión al riesgo, penalizando los costos variables no deseados, definidos como los costos variables por encima del costo esperado.

Para distintos valores de *w* tenderemos diferentes pesos para el término de aversión a los costos por encima del esperado, lo cual nos puede guiar a soluciones más baratas con respecto al costo de transportación esperado, pero al mismo tiempo le resta importancia o ponderación al término del costo del diseño y transportación de productos. Al problema (1) - (4) lo referiremos como DRRCM (Diseño Robusto de Red Capacitada Multiproducto).

CAPÍTULO 4. METODOLOGÍA

4.1 Introducción.

Este apartado abordará la metodología seleccionada para resolver nuestro problema de diseño de red capacitada multiproducto sujeta a incertidumbre en parámetros clave. Esta metodología es un GRASP basado en uno similar desarrollado para la versión determinista de este problema (De Alba, 2004 [16]).

En la presente obra se desarrollaron tres versiones de la metodología GRASP, con el fin de probarlas y compararlas entre sí. Las versiones desarrolladas son:

- Un "GRASP con memoria" ($GRASP^{M}$) que incluye, además de las funciones de valor y selección aleatoria propias del GRASP tradicional y reportadas en la mayoría de las implementaciones (Arráiz et al., 2001 [1]; Chardaire, McKeown y Maki, 2001 [6]; Binato et al., 2002 [5]), estructuras de memoria que permiten analizar y retener las características inherentes a las buenas soluciones generadas previamente. La versión con memoria del GRASP trabaja con un escenario ficticio definido más adelante, creado a partir del problema original.
- Un "GRASP Robusto" ($GRASP^{R}$) con las características del GRASP con estructuras de memoria, pero con una función de evaluación o de valor de tipo robusta, descrita más adelante (esta versión trabaja con la totalidad de los escenarios, y no solo con el escenario ficticio).
- Y un "GRASP Ajustado" ($GRASP^{A}$) que recorre los escenarios uno a uno, en busca de diseños que requieran menores capacidades en la red, con el respectivo ahorro en costo.

Teniendo en cuenta que existen diversos escenarios futuros posibles para la demanda de cada producto, y que se desean construir diseños que sean factibles para cualquier escenario que se presente, tal que la demanda de cada producto bajo cualquier

escenario pueda ser transportada, se creará un escenario ficticio con el cual trabajarán las dos primeras versiones de GRASP desarrolladas.

Este escenario ficticio tendrá los siguientes parámetros:

- A cada arista se le asociará un costo fijo igual a f_{ij}.
- A cada arista se le asociará una capacidad igual a u_{ij} (recordemos que ni los costos fijos ni las capacidades son sujetas a incertidumbre en nuestro trabajo).
- La demanda de cada producto se definirá como:

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

- Y los costos variables serán:

$$C_{ijk} = E(\, c_{ijk}(s)\,),$$

donde $E(c_{ijk}(s))$ se calcula como:

$$E(c_{ijk}(s)) = \sum_{s \in S} p_s (c_{ijk}(s)) \qquad \forall (i,j) \in A,\ k \in K$$

El problema de diseño de redes sobre este escenario ficticio se modela de la siguiente forma:

$$\min \sum_{k \in K} \sum_{\{i,j\} \in E} C_{ijk} x_{ijk} + \sum_{\{i,j\} \in A} f_{ij} y_{ij} \tag{5}$$

$$\sum_{\{j:(i,j) \in A\}} x_{ijk} - \sum_{\{j:(j,i) \in A\}} x_{jik} = \begin{cases} D_k & \text{si } i = \text{O(k)} \\ -D_k & \text{si } i = \text{D(k)} \\ 0 & \text{de otro modo} \end{cases} \qquad \forall\, k \in K,\ \forall\, i \in N \tag{6}$$

$$\sum_{k \in K} \left(x_{ijk} + x_{jik} \right) \le u_{ij} y_{ij} \qquad \forall \{i,j\} \in E \tag{7}$$

$$x_{ijk} \ge 0 \qquad \forall\, k \in K,\ \forall (i,j) \in A \tag{8}$$

$$y_{ij} \in \{0,1\} \qquad \forall\, \{i,j\} \in E$$

Al problema (5) - (8) lo referiremos como DFRCM (Diseño Ficticio de Red Capacitada Multiproducto). El escenario ficticio será de utilidad en pasos siguientes, para simplificar los pasos iniciales de nuestro GRASP, como se expondrá más adelante).

4.2 Representación de una solución.

El diseño e implementación de un GRASP, metaheurística que será utilizada para la búsqueda de soluciones, exige una representación computacional amigable que favorezca un tratamiento eficiente, así como una elección adecuada de los elementos que irán conformando paso a paso, una solución cualquiera al problema propuesto. En este caso, esos elementos serán rutas entre el origen y destino de cada producto.

En el problema bajo estudio, una solución se representará como elementos independientes (un elemento por producto), llamados *bloques de rutas*. Para cada producto, cuya demanda será transportada en la red, se encuentra un conjunto de rutas mediante las cuales dicha demanda pueda ser transportada en su totalidad. A este conjunto de rutas se le llamará "bloque de rutas". Existe por lo tanto un bloque de rutas para cada producto y cada bloque de rutas puede estar compuesto de una o más rutas, las cuales se irán seleccionando de entre las rutas "más cortas" encontradas para el producto que se analiza. Si bien todos los productos pueden compartir la capacidad de las aristas, los bloques de rutas serán considerados y tratados en forma independiente uno de otro. Una representación gráfica está ejemplificada en la figura 1.

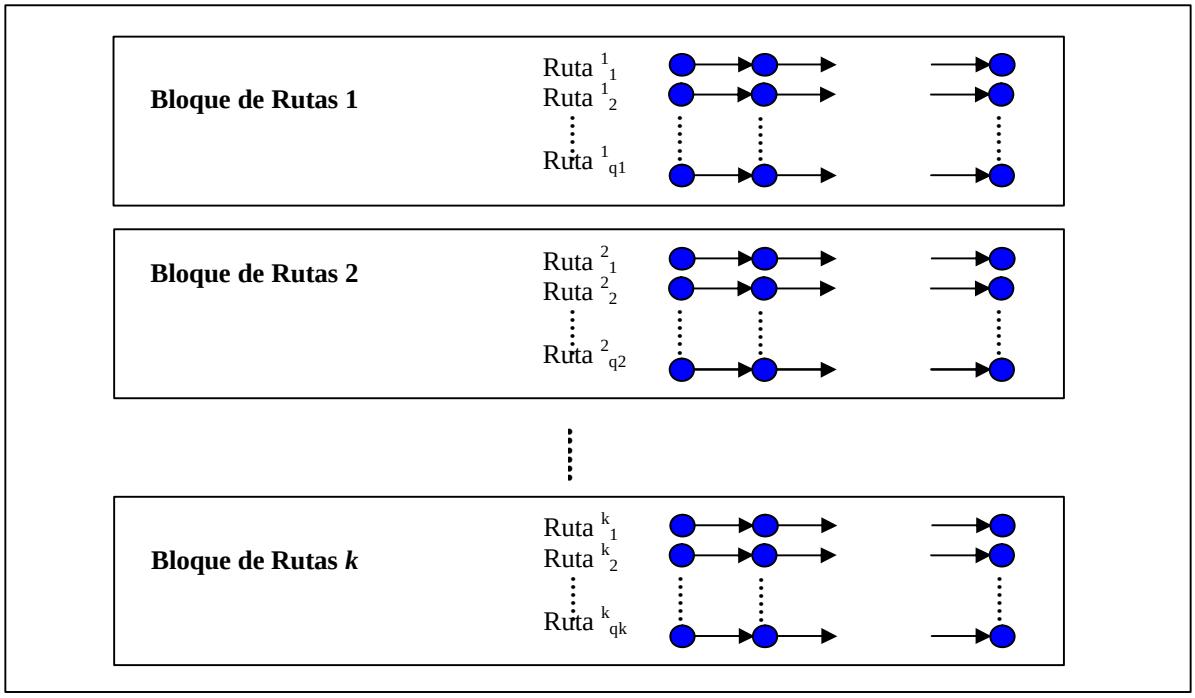


Figura 1. Representación de una solución

4.3 Obtención de las *q* Rutas Más Cortas.

Para poder formar una solución al problema DRRCM, es necesario disponer de un conjunto de rutas a partir de las cuales se irán conformando los bloques que transportarán la demanda de cada producto. Para lograr lo anterior se encontrarán *q* rutas cortas entre cada par origen-destino en la red, donde *q* es un parámetro dado durante la implementación (Lawler, 1972 [54]).

De entre este conjunto de rutas cortas, serán seleccionadas por GRASP la o las más promisorias para conformar los bloques de rutas que transportarán la demanda de cada producto.

Para definir una ruta más corta, debemos primero definir longitudes de arco que nos permitan comparar una ruta con otras y decir cuál es más corta. Para lograr esto, se definieron longitudes de arcos de tres tipos, y para cada tipo se generaron igual número de rutas más cortas, con el fin de darles igual importancia a cada tipo de longitud.

Cabe destacar que todas las operaciones realizadas para la generación de las *q* rutas más cortas se hacen sobre el escenario ficticio definido líneas arriba, utilizar este escenario vuelve muy sencilla la generación de las rutas más cortas, además de que no hay una gran variación en cuanto a las rutas determinadas por algún otro escenario, según lo observado durante las pruebas realizadas.

Los tipos de longitudes se definen como sigue:

1. $lenght_{ij}^{k} = \left(C_{ijk} + F_{ij}\right)\left(1 + \left|\frac{\left(u_{ij} - D^{k}\right)}{D^{k}}\right|\right)$

2. $lenght_{ij}^{k} = \left(C_{ijk} + F_{ij}\right)\left(1 + \left|\frac{\left(u_{ij} - \sum_{h \in H_{ij}} D^{h}\right)}{\sum_{h \in H_{ij}} D^{h}}\right|\right)$

3. $lenght_{ij}^{k} = \left(C_{ijk} + F_{ij}\right)\left(1 + \frac{F_{ij}}{F'}\right)$

En la longitud de tipo 1, se penaliza la desviación de la demanda del producto k respecto a la capacidad de la arista correspondiente a ese arco. La finalidad de esta longitud es hacer atractivos aquellos arcos cuya capacidad es empleada efectivamente, es decir sin desperdicio, al intentar transportar la demanda de un producto a través de él. Debido a esto, un arco cuya capacidad sea igual a la demanda de un producto no es penalizado.

En la longitud de tipo 2, H_{ij} es el subconjunto de productos que podrían compartir junto con el producto k la arista $\{i, j\}$ (esto es el arco (i,j) o (j,i)) en aquellas rutas encontradas con la longitud tipo 1. En otras palabras, el conjunto H_{ij} lo forman el producto k junto a los productos que utilizaron la arista $\{i, j\}$ en las rutas tipo 1. En este caso, la penalización se hace sobre la desviación de la capacidad de la arista respecto a la suma de las demandas de aquellos productos que potencialmente pudieran ser transportados a través de ella, La finalidad de esta longitud es hacer atractivas aquellas aristas cuyo costo fijo sea distribuido lo mejor posible entre todos los productos que la usen.

Finalmente, en la longitud de tipo 3, se ha considerado un factor de penalización para introducir cierta diversificación en el conjunto de rutas ya generadas. En este caso, F_{ij} representa la frecuencia de la arista $\{i, j\}$ en las rutas previamente encontradas, esto es, el número de veces que la arista $\{i, j\}$ ha aparecido en algunas otras rutas. F' representa la frecuencia máxima de aparición de una arista en las rutas ya encontradas.

Una vez determinadas las rutas cortas con cada tipo de longitud (q/3 rutas de cada tipo, donde q es un parámetro de implementación), estas rutas se incorporarán a una lista de las q rutas más cortas determinadas para cada producto k, la cual se denotará LP^k.

Es importante señalar que la longitud de un arco puede variar de producto a producto. Las rutas obtenidas con las longitudes de tipos 1 y 2 se pueden considerar como rutas atractivas, mientras que las rutas encontradas con la longitud de tipo 3 pueden considerarse diversas respecto a las anteriores.

4.4 Fase Constructiva.

Las primeras dos versiones de GRASP, descritas más adelante, están basadas en un trabajo previo de diseño de red capacitada multiproducto determinista (De Alba, 2004 [18]).

El algoritmo que se expone a continuación es el seguido por el $GRASP^M$ y el robusto. El $GRASP^A$ es un poco distinto, y se describirá a detalle en el punto correspondiente.

De manera general, la fase constructiva de las primeras dos versiones GRASP puede describirse como sigue:

1. Ordenamiento de los productos. Los productos son acomodados aleatoriamente o según algún orden preestablecido.
2. Selección de un producto *k*. Los productos se van seleccionando según el orden en el que fueron acomodados en el paso anterior y se ejecutan los pasos 3 al 6 para cada producto.
3. Evaluación de rutas. A cada ruta contenida en LP^k (conjunto de rutas más cortas para el producto *k* que se está analizando), se le asigna una evaluación.
4. Construcción de la lista restringida de candidatos (LRC^k). Se forma una lista restringida de candidatos para cada producto *k* (LRC^k) tomando las rutas mejor evaluadas (y capaces de transportar al menos una unidad de flujo) de LP^k.
5. Selección de una ruta. Se selecciona una ruta de manera aleatoria de la LRC^k, o bien de manera ponderada según su evaluación. Se actualizan las capacidades residuales, al igual que los costos fijos residuales, para cada arista en esa ruta. La demanda residual del producto que se analiza es actualizada de igual manera. Si no hubiera más rutas disponibles y si la demanda residual fuera mayor a cero, entonces no ha podido construirse una solución factible y el proceso se repite desde el paso 1.
6. Adición de una ruta al bloque de rutas. A medida que las rutas van siendo seleccionadas, se van incorporando al Bloque de Rutas correspondiente al producto que se trate. Se repiten los pasos 3 al 6 hasta que la demanda residual es cero o no hay más rutas.

Al final del procedimiento, tendremos una solución σ^0 formada por un conjunto de bloques de rutas (un bloque para cada producto) capaces cada uno de ellos de transportar la demanda completa de cada producto. Este proceso puede repetirse tantas veces como sea necesario para producir la cantidad de soluciones que se desee.

Como se comentó previamente, este algoritmo se empleará en la fase constructiva del $GRASP^M$ y el $GRASP^R$. Las diferencias entre ambos GRASP estarán dadas principalmente por las funciones de evaluación de rutas y el problema "base" al que se aplican. El GRASP con memoria se aplica al problema DFRCM, resolviendo por tanto un problema determinista con un solo escenario de datos: el ficticio. Por su parte, el $GRASP^R$ trabaja sobre el problema DRRCM, considerando todos los escenarios posibles de datos.

La tercera versión del GRASP, el $GRASP^A$ también resolverá el problema DRRCM considerando todos los escenarios posibles de datos, sin embargo el algoritmo de la fase constructiva difiere un poco del que acabamos de describir y será especificado más adelante.

4.4.1 GRASP^{M}.

La primera versión de GRASP, contrario a lo que ofrecen las implementaciones tradicionales, incorpora mecanismos de memoria o aprendizaje al proceso de creación de soluciones para obtener información útil que ayude a la generación de futuras soluciones.

Para encontrar soluciones (diseños) al problema DRRCM, este GRASP trabajará con un solo escenario de datos, el escenario ficticio definido en 4.1. Esto es, se resolverá el problema DFRCM. Las soluciones (diseños) a este problema serán soluciones factibles para cualquier escenario de datos que se presente en el problema DRRCM.

Como habíamos definido anteriormente, en un algoritmo miope por lo general hay funciones que evalúan el beneficio de incorporar una determinada selección en una solución parcial, es decir, nos dicen que tan “bueno” sería añadir cierto elemento a la solución parcialmente construida.

En esta versión, las selecciones del GRASP dependerán de las rutas que van del origen al destino de cada producto, trabajando únicamente sobre el escenario ficticio creado, es decir, el definido con la demanda máxima para cada producto sobre todos los escenarios y los costos variables esperados sobre todos los escenarios. De este conjunto de rutas se seleccionarán las que formarán parte del bloque de rutas para el transporte de la demanda completa de cada producto (ver sección 4.2), por lo que deben recibir una evaluación que exprese qué tan conveniente resulta añadir esa ruta a la solución.

Sin embargo, para que la evaluación considerada constituya una medida adaptativa es necesario que los valores determinados por la función de evaluación sean actualizados, con la finalidad de incorporar la nueva información recolectada después de que cada ruta es seleccionada. En la sección 4.3 se mencionó que era necesario determinar un conjunto de rutas “más cortas”, y es a partir de este conjunto que formaremos una lista restringida de candidatos, con una probabilidad de selección asociada para cada ruta, de donde se irán seleccionando estas últimas para formar el bloque de rutas que transportará la demanda para cada producto.

Como deseamos generar una colección de soluciones diversas, la selección de rutas a pertenecer al bloque de rutas para cada producto se hace de forma totalmente aleatoria, a partir de la lista de candidatos disponible y de las probabilidades asociadas a cada uno de sus elementos.

De manera adicional, y con el fin de lograr los objetivos de aprendizaje, se implementaron los conceptos de variables fuertemente determinadas y variables consistentes. Las variables fuertemente determinadas dependen de un conjunto de soluciones élite cuyos valores no pueden cambiar sin degradar substancialmente el valor de la función objetivo. Una variable consistente es aquélla que recibe un valor particular en una porción significativa de buenas soluciones. Tal como fue propuesto por Fleurent y Glover (1999 [25]), se genera un conjunto s_{sol} de r soluciones élite y se actualiza (si procede) cada vez que se genera una nueva solución. A medida que nuevas soluciones factibles vayan siendo generadas, éstas pueden reemplazar a las soluciones en el conjunto s_{sol}.

El GRASP^{M} tiene las características básicas de cualquier GRASP, como lo son la adaptabilidad y voracidad mencionadas anteriormente, en adición de los procedimientos de memoria definidos para el aprendizaje.

A fin de mantener un proceso sencillo en la manipulación y generación de soluciones, el GRASP^{M} trabaja con un escenario ficticio generado a partir de los escenarios originales del problema completo. Este escenario ficticio, como recordamos se forma con las demandas máximas para cada producto y los costos variables esperados, a través de todos los escenarios.

4.4.1.1 Evaluación de Rutas en el GRASP^{M}.

La función de evaluación de rutas que se utiliza, mide el beneficio de incorporar una ruta específica al bloque que se está formando para cada producto. Esta función que mide el beneficio de incluir una ruta p en la solución parcialmente construida $s*$ la hemos denominado *value(s*, ruta p).* Mientras mayor sea el valor de esta función mejor será la bondad de la ruta analizada.

La función *value* está definida en relación a otras medidas, como lo son la función *C(bloque k, ruta p)* que nos muestra una evaluación del costo de asignación de una ruta *p* al bloque de rutas del producto *k*. Esta evaluación de costo, será una función auxiliar para nuestra implementación

$$C(bloque\ k, ruta\ p) = \sum_{(i,j)\in ruta\ p} \left(C_{ijk} D'^{k} + f'_{ij}\right)\left(1 + \left|\frac{\left(u'_{ij} - D'^{k}\right)}{D'^{k}}\right|\right)$$

Aquí debemos observar lo siguiente:

- Ruta *p* se refiere a cualquier ruta desde O(*k*) a D(*k*);
- f'_{ij} se refiere al costo fijo residual (si la arista {*i*,*j*} no ha sido empleada en rutas anteriores del diseño, el costo fijo residual será f_{ij}, si ya ha sido utilizada el costo fijo residual será cero). Representa el costo de utilizar la arista {*i*,*j*} la primera vez que esta arista aparece en el diseño de la red;
- u'_{ij} es la capacidad residual de la arista {*i*,*j*} (es decir, la capacidad no empleada en la arista conforme se hace circular algún producto a través de ella);
- D'^{k} es la demanda residual del producto *k* (esto es, la demanda que falta ser transportada para el producto *k* en un punto determinado en la creación de la solución). Recordemos que la demanda con la que se trabaja en este caso es la demanda del producto *k* en el escenario ficticio, es decir, la demanda máxima para cada producto encontrada sobre todos los escenarios.
- C_{ijk} es el costo variable por unidad de flujo del producto *k* cuando usa el arco (*i*,*j*). Recordemos que este es el costo variable definido en el escenario ficticio, esto es, el costo esperado encontrado sobre todos los escenarios de nuestro problema original.

En la función de costo *C*(*bloque k, ruta p*) existe un término de penalización hacia las rutas que tengan una capacidad residual sensiblemente distinta a la demanda residual del producto analizado en ese momento, este término es:

$$\left(1 + \left|\frac{\left(u'_{ij} - D'^{k}\right)}{D'^{k}}\right|\right)$$

El objetivo de esta penalización es tener como bien evaluadas las rutas cuyas capacidades residuales sean lo mejor aprovechadas por las demandas residuales de cada

producto, evitando añadir rutas baratas en términos unitarios de capacidad, pero mal aprovechadas, ya sea por tener poca capacidad cuando necesitemos mucha, o viceversa.

Si la ruta *p* fuera asignada al bloque de rutas del producto *k* en la solución parcial s^*, el valor de la evaluación de costo auxiliar de la solución sería el siguiente:

$$T^k(s^*, ruta\ p) = \sum_{m=1}^{k-1}\sum_{l=1}^{b^m} C(bloque\ m, ruta\ l) + \sum_{l=1}^{p} C(bloque\ k, ruta\ l)$$

b^m es el número de rutas incluidas en el bloque del producto *m*.

El mínimo costo en la lista de rutas disponible será definido como:

$$best\_cost(LP^k) = min\ \{\ T^k(s^*, ruta\ t) : ruta\ t\ \hat{I}\ LP^k\ \}$$

Es entonces que el costo de cada ruta se normaliza considerando $best\_cost(LP^k)$ de forma tal que el valor de la función definida para cada ruta *p* que pertenezca a esta lista se calculará como sigue:

$$value(s^*, ruta\ p) = best\_cost(LP^k) / T^k(s^*, ruta\ p)$$

Esta evaluación de rutas que mide el beneficio de incorporar una ruta al bloque de rutas que se forma para cada producto se empleará para la construcción del conjunto s_{sol} de soluciones élite.

Como se ha mencionado, con la finalidad de aprender de las soluciones generadas con antelación, y de las rutas previamente seleccionadas en la solución actual, es necesario tener una medida de las características fuertemente determinadas y consistentes de la selección de la ruta que será incluida en la solución s^*. Por ello definiremos una nueva función, $intensity(s^*, ruta\ p)$, que nos permita retener las características de las buenas soluciones generadas con anterioridad. A mayores valores de esta función mayor es la frecuencia con que la ruta *p* aparece en el conjunto s_{sol} de soluciones élite.

Con respecto a las soluciones incluidas en el conjunto élite s_{sol}, el costo real de una solución, medida por el valor de su función objetivo, será denotado $Cost(s)$ y podremos normalizar el costo de cada solución dentro del conjunto élite s_{sol} con la función:

$$Value(s) = best\_cost(s_{sol}) / Cost(s)$$

Donde $best\_cost(s_{sol}) = min\ \{Cost(s) : s\ \hat{I}\ s_{sol}\ \}$

Value(s) será entonces, una medida de la bondad de las rutas en s_{sol}. La función de intensidad por asignar una ruta *p* a la nueva solución creada s^*, será ahora:

$$intensity(s^*, ruta\ p) = \sum_{s \in S_{sol}: ruta\ p \in bloque\ k} Value(s)$$

Por último, la función encargada de combinar los valores de costo e intensidad de una selección determinada se definirá como una función creciente de sus argumentos:

$$E(s^*, ruta\ p) = ?\ value(s^*, ruta\ p) + intensity(s^*, ruta\ p)$$

Diferentes valores de l causarán mayor énfasis en el término de diversidad o intensidad, lo cual a su vez, guiará la selección de rutas. Bajos valores de l incluyendo el cero, causarán más énfasis en la intensidad, favoreciendo la selección de rutas que ya han aparecido en soluciones de alta calidad. Altos valores de l favorecerán la selección de rutas altamente evaluadas sin importar su contribución en el conjunto de soluciones élite.

En resumen, las siguientes funciones han sido definidas:

- *C*(*bloque k, ruta p*), la función que indica una evaluación del costo de asignación de la ruta *p* al bloque de rutas del producto *k* ;
- $T^k(s^*, ruta\ p)$ que representa una evaluación del costo de la solución parcial si se asignara al bloque del producto *k* la ruta *p* en una solución s^* parcialmente construida;
- *value*(s^*, *ruta p*), que es la evaluación del costo T^k de cada ruta normalizado con respecto al mejor costo encontrado en su lista LP^k;
- *intensity*(s^*, *ruta p*), mide la frecuencia con que la ruta *p* aparece en el conjunto s_{sol} de soluciones élite;
- *Value*(s), es el costo de una solución medida por el valor de su función objetivo (*Cost*(s)), normalizado con respecto al mejor costo encontrado en s_{sol} (*best_cost*(s_{sol})) y,
- *E*(s^*, *ruta p*), la función encargada de combinar los valores de costo e intensidad de una selección determinada.

4.4.1.2 Construcción de la LRC^k y Selección de Rutas.

Las rutas de LP^k se ordenan de forma no creciente con respecto a su función de evaluación *value*(s^*, *ruta p*) definida previamente, con la finalidad de crear una LRC^k (Lista Restringida de Candidatos del Producto *k*). La cardinalidad de esta lista para la versión con memoria del GRASP se considerará fija.

Para seleccionar una ruta a partir de la LRC^k en la versión con memoria de GRASP, se establece una probabilidad de selección de la ruta *p*, en donde tendrá mayor probabilidad de selección un elemento miembro de la LRC^k con un valor alto de $E(s^*, ruta\ p)$.

Para cada ruta de la LRC^k su probabilidad de ser seleccionada está dada por:

$$P(s^*, ruta\ p) = \frac{E(s^*, ruta\ p)}{\sum_{ruta\ l \in LRC^k} E(s^*, ruta\ l)}$$

Debemos tener presente que $E(s^*, ruta\ p)$ es la función encargada de evaluar una solución en cuanto a calidad e intensidad en la versión del GRASP^M.

Habiendo seleccionado una ruta para formar parte del bloque de rutas para el producto analizado, calcularemos la capacidad de la ruta, que es el máximo flujo posible de enviar desde el origen hasta el destino de esta ruta. La capacidad de una ruta, será la capacidad del arco con menor capacidad residual en toda la ruta.

Una vez encontrada la capacidad de la ruta se mandará tanto flujo como sea posible hasta saturar esta ruta. En el caso de que la capacidad de la ruta sea menor o igual a la demanda del producto analizado el proceso de enviar flujo dejará la capacidad residual en cero de al menos un arco en la ruta. En caso contrario, el flujo enviado será el necesario para completar la demanda del producto y se continúa este proceso con el siguiente producto.

Cada vez que se añada una ruta a un bloque de rutas de un producto, se actualizarán los valores de la función de evaluación de las rutas aún no seleccionadas, así como la LRC^k.

4.4.1.3 Actualización del Conjunto Élite s_{sol}.

El conjunto de soluciones élite s_{sol} inicialmente contendrá *r* soluciones de costo infinito, con lo que las primeras *r* soluciones encontradas a nuestro problema formarán parte de s_{sol}. Para que una solución generada por GRASP tome el lugar de alguna de las soluciones en el conjunto s_{sol}, deberá tener mejor evaluación que la mejor solución en s_{sol} o ser mejor que la peor solución de este mismo conjunto y al mismo tiempo estar suficientemente distante del resto de las soluciones.

La distancia entre dos soluciones se determina con la siguiente función:

$$d(s^1, s^2) = \frac{\sum_{k \in K} \sum_{p_j \in LP^k} \left| t_j^1 - t_j^2 \right|}{h_1 + h_2}$$

En donde:

- LP^k es la lista de las *q* rutas más cortas determinadas para cada producto *k*;
- t^i es el vector característico de la solución s^i correspondiente a todas las rutas encontradas para cada producto, es decir, a cada conjunto LP^k. Si la solución s^i usa la ruta p_j, el valor de $t^i_j = 1$, en caso contrario $t^i_j = 0$. El vector t^i tendrá $q \left| \mathrm{K} \right|$ componentes;
- h_i es la cantidad de rutas empleadas en la solución *i*.

La función $d(s^1, s^2)$ evaluará a 0 si la solución s^1 es igual a la s^2 ó 1 si ambas soluciones son enteramente distintas, es decir si no tienen ninguna ruta en común. Cualquier caso intermedio, evaluará a un valor entre 0 y 1.

Como se había mencionado, una solución recién creada s puede formar parte del conjunto élite s_{sol}, pero para determinarlo es necesario comparar su evaluación de función objetivo y que sea mejor que la del mejor elemento del conjunto s_{sol} actual. En este caso, la solución recién creada formará parte del conjunto élite del cual será eliminada la solución peor evaluada.

En el caso de que la nueva solución no sea mejor que la mejor solución de s_{sol}, pero que sí sea mejor que la peor solución del conjunto, habrá que determinar la distancia d de s a todas las soluciones en el élite y promediar este valor. Si este

promedio es mayor que un cierto umbral establecido en la implementación (umbral de sustitución en el conjunto élite), la nueva solución tomará el lugar de la peor solución del conjunto élite s_{sol}.

4.4.2 GRASP^{R}.

La segunda versión de GRASP que es desarrollada en este trabajo es la que denominaremos robusta. Esta versión incorpora una función de evaluación de tipo robusta y en lugar de trabajar simplemente con un escenario ficticio de demanda máxima y costos variables esperados con respecto a los escenarios de nuestro problema original, evalúa la solución parcialmente creada considerando todos los escenarios a la vez. Es decir, este GRASP se aplica en sí al problema DRRCM, si bien hay que aclarar que las *q* rutas obtenidas para cada producto se basan en los parámetros del escenario ficticio.

El GRASP^{R} contiene las características de memoria de la versión anterior, ya que estos mecanismos de aprendizaje guiado a través de la creación de soluciones, demostraron tener buenos resultados.

4.4.2.1 Evaluación de Rutas en el GRASP^{R}.

Si bien la mayoría de las funciones empleadas en esta versión son prácticamente las mismas que las que usa el GRASP^{M}, existe una diferencia muy importante en la función de evaluación *C*(*bloque k, ruta p*) función que nos da una estimación del costo de asignación de una ruta *p* al bloque de rutas del producto *k.* En este caso, se propone la siguiente función:

$$C(bloque\ k, ruta\ p) = \sum_{s \in S} p_s \left[\left(\sum_{(i,j) \in\ ruta\ p} \left(z_{ijk}^{s} + f'_{ij} \right) \right) \left(1 + \left| \frac{\left(u'_{ij} - d'^{s}_{k} \right)}{d'^{s}_{k}} \right| \right) \right] + w^{0}$$

Donde:

$$w^{0} = w \sqrt{\frac{\sum_{s \in S^{+}} p_s \left(z_{ijk}^{s} - E(z_{ijk}^{s}) \right)^{2}}{\sum_{s \in S^{+}} p_s}}$$

$$z_{ijk}^{s} = c_{ijk}^{s} d'^{s}_{k}$$

$$E(z_{ijk}^s) = \sum_{s \in S} p_s z_{ijk}^s$$

$$S^+ = \{s : z_{ijk}^s - E(z_{ijk}^s) \geq 0\}$$

ω = Ponderación para la penalización

Aclaremos que:

- Ruta *p* se refiere a cualquier ruta desde O(*k*) a D(*k*);
- f'_{ij} se refiere al costo fijo residual (si la arista {*i*,*j*} no ha sido empleada en rutas anteriores del diseño, el costo fijo residual será f_{ij}, ó 0 en caso contrario). Representa el costo de utilizar la arista {*i*,*j*} la primera vez que esta arista aparece en el diseño de la red;
- u'_{ij} es la capacidad residual de la arista {*i*,*j*} (es decir, la capacidad que va quedando en la arista conforme se hace circular algún producto a través de ella);
- $d_k^{'s}$ es la demanda residual del producto *k* en el escenario *s* (esto es, la demanda que falta ser transportada para el producto *k* en un punto determinado en la creación de la solución).
- c^s_{ijk} es el costo variable por unidad de flujo del producto *k* en el escenario *s* cuando usa el arco (*i*,*j*).

Ahora conviene señalar las funciones adicionales que empleamos en esta versión robusta. Del GRASP^M se usarán las siguientes funciones definidas en el apartado 4.4.1.1:

- $T^k(s^*, ruta\ p)$ que representa una estimación del costo de haber asignado al producto *k* una ruta *p* en una solución s^* parcialmente construida y,
- *value*(s^*, *ruta p)*, que es el costo de cada ruta normalizado con respecto al mejor costo encontrado en LP^k.
- *Value*(s), es el costo de cada solución normalizada con respecto a la solución mejor evaluada con respecto a su función objetivo, que para el caso del GRASP^R será la función objetivo robusta definida en 4.4.2.1;
- *intensity*(s^*, *ruta p*), es una medida de las características fuertemente determinadas y consistentes de la selección de la ruta que será incluida en la solución s^*

- $E(s^*, ruta\ p)$, la función encargada de combinar los valores de costo e intensidad de la solución creada en cuestión.

Todas las observaciones especificadas para la versión de GRASP^M aplican en la versión robusta. Es necesario tener presente que de manera adicional a una evaluación de costo, tenemos un término de penalización (ω^0) en nuestra función objetivo, que busca hacer atractivas las soluciones que presentan buen desempeño para todos los escenarios y no solo para algunos de ellos.

4.4.2.2 Construcción de la LRC^k y Selección de Rutas.

El GRASP de características robustas construye la LRC^k de la misma manera que lo hace la versión con memoria, es decir, ordenando las rutas de LP^k con respecto a su función de evaluación y formando una LRC^k. La cardinalidad en esta versión del GRASP se considerará también fija, para incluir sólo las rutas mejor evaluadas.

Para la selección de las rutas que se incorporarán a la solución que se está creando se establece una probabilidad de selección de la ruta *p*, que otorga mejor oportunidad de selección a un elemento miembro de la LRC^k con un valor alto de $E(s^*, ruta\ p)$. La fórmula para determinar dicha probabilidad se define en la sección 4.4.1.2.

4.4.2.3 Actualización del Conjunto Élite s_{sol}.

La actualización del conjunto élite en la versión robusta del GRASP no es distinta de la definida para la versión con memoria; se comienza con *r* soluciones "nulas" de costo infinito como miembros del conjunto s_{sol}.

Las primeras *r* soluciones creadas formarán parte del conjunto élite pero para que alguna de las subsecuentes soluciones generadas por GRASP tome el lugar de una de las soluciones en el conjunto s_{sol}, deberá tener mejor evaluación que la mejor solución en s_{sol} o ser mejor que la peor solución de este mismo conjunto y al mismo tiempo estar suficientemente distante del resto de las soluciones. La función que mide la distancia entre dos soluciones es la de $d(s^1, s^2)$ analizada anteriormente.

4.4.3 GRASP^{A}.

Como ya se había mencionado, la versión "ajustada" del GRASP resuelve también el problema DRRCM considerando todos los escenarios posibles de datos. Esta versión recorre cada uno de los escenarios originales de nuestro problema, con la esperanza de encontrar un diseño más barato, en comparación a diseños que observan sólo el escenario ficticio (GRASP^{M}), o que consideran todos los escenarios simultáneamente (GRASP^{R}), que si bien nos garantizan factibilidad en el total de los escenarios, hay un costo extra incurrido por usar esta comodidad. Además, el escenario ficticio, realmente nunca ocurre, por lo cual nos estamos preparando para un evento que no llegará y desperdiciando recursos preciados.

Observando este hecho, se modificó el algoritmo del GRASP para el problema de DRRCM, para considerar lo mencionado, y dando como resultado el GRASP^{A}:

1. Ordenamiento de los escenarios. Los escenarios son acomodados aleatoriamente o según algún orden definido, y seleccionamos uno de ellos según su orden.
2. Ordenamiento de los productos. Los productos son acomodados aleatoriamente o según algún orden definido.
3. Selección de un producto *k*. Los productos se van seleccionando según el orden en el que fueron acomodados en el paso anterior y se ejecutan los pasos 4 al 8 para cada producto.
4. Evaluación de rutas. A cada ruta contenida en LP^{k} (conjunto de rutas más cortas para el producto *k* que se está analizando), se le asigna una evaluación.
5. Construcción de la lista restringida de candidatos (LRC^{k}). Se forma una lista restringida de candidatos (LRC^{k}) tomando las rutas mejor evaluadas (y capaces de transportar al menos una unidad de flujo) de LP^{k}.
6. Selección de una ruta. Se selecciona una ruta de manera aleatoria de la LRC^{k}, o bien de manera ponderada según su evaluación. Se actualizan las capacidades residuales, al igual que los costos fijos residuales, para cada arista en esa ruta. La demanda residual del producto que se analiza es actualizada de igual manera. Si no hubiera más rutas disponibles y si la demanda residual fuera mayor a cero,

entonces no ha podido construirse una solución factible y el proceso se repite desde el paso 2.

7. Adición de una ruta al bloque de rutas. A medida que las rutas van siendo seleccionadas, se van incorporando al Bloque de Rutas correspondiente al producto que se trate. Se repiten los pasos 4 al 7 hasta que la demanda residual es cero o no hay más rutas.
8. Continuar con escenarios no analizados. Mientras queden escenarios sin analizar, continuaremos con el siguiente escenario según su orden y comprobaremos que el diseño construido hasta este momento pueda transportar la demanda del siguiente escenario:
    a. Si es posible enviar toda la demanda del siguiente escenario, con el diseño construido hasta el momento, reiniciar el paso 8.
    b. En caso contrario, obtendremos una nueva ruta más corta con Costos Fijos y Capacidades de Aristas residuales para el producto en el que se haya detectado infactibilidad, y se añadirá al diseño. Reiniciar el paso 8.
9. Diseño final. Una vez observados todos los escenarios del problema original, tendremos un diseño factible a todos los escenarios con un costo al menos igual, y quizá menor que si hubiéramos analizado únicamente el escenario ficticio de nuestro problema.

Se decidió utilizar las mismas rutas cortas encontradas mediante un escenario ficticio (ver punto 4.3) en primer lugar, por ser relativamente fáciles de obtener; y en segundo lugar para poder comparar las diversas versiones GRASP en igualdad de circunstancias.

4.4.3.1 Evaluación de Rutas en el GRASP^{A}.

La evaluación de las rutas para esta versión de GRASP, es idéntica a la evaluación de rutas para el GRASP^{M} (ver sección 4.4.2.1) incluyendo la estimación de costo C(*bloque k, ruta p*), y las evaluaciones adicionales definidas para esa versión GRASP.

4.4.3.2 Construcción de la LRC^k y Selección de Rutas en el $GRASP^A$.

La construcción de la LRC^k para el $GRASP^A$ es igual a los GRASP anteriores, puede verse el apartado 4.4.2.2 para mayor detalle, y la sección 4.4.1.2 para observar la definición de la fórmula que determina la probabilidad de selección de una ruta *p* como elemento miembro de la LRC^k.

4.4.3.3 Construcción de Rutas Adicionales en el $GRASP^A$.

Un mecanismo que no presentan las versiones GRASP anteriores consiste en una construcción de rutas adicionales conforme se van necesitando. Se había mencionado que para la versión de $GRASP^A$, se utilizaban las rutas cortas obtenidas inicialmente (punto 4.3) para comenzar a construir una solución factible, al menos para uno de los escenarios potenciales, y se analizaba ese diseño respecto a los demás escenarios para comprobar factibilidad robusta en la solución parcialmente construida.

Para ese primer paso, se usan las rutas cortas iniciales, que se suponen buenas para nuestro diseño. Sin embargo, conforme avanzamos en la construcción de una solución, la medida que les otorgó bondad en un principio (evaluaciones del punto 4.3), rápidamente pierde actualidad, y una ruta buena al inicio de la construcción, deja de serlo cuando tenemos un diseño parcial.

Este razonamiento nos hace desear tener nuevas rutas cortas en cada paso de la construcción de una solución y no contar solo con las rutas originales. Es por esto que, cada vez que es necesaria una ruta adicional para recuperar factibilidad cuando se analiza el diseño parcial (factible para algún escenario, pero infactible en algún punto para otro) se calcula una nueva ruta y se añade para lograr la factibilidad robusta deseada.

Las nuevas rutas cortas se calculan con los costos fijos residuales y costos variables esperados para las aristas que aún poseen capacidad residual, con la siguiente fórmula:

$$lenght_{ij}^{k} = \left(C_{ijk} + F'_{ij}\right)\left(1 + \left|\frac{\left(u'_{ij} - d'^{k}\right)}{d'^{k}}\right|\right)$$

En donde:

C_{ijk} Es el valor esperado del costo variable de la arista $\{i,j\}$ para el producto k a través de todos los escenarios.

F'_{ij} Es el Costo Fijo residual de la arista $\{i,j\}$.

u'_{ij} Es la capacidad residual de la arista $\{i,j\}$.

d'^{k} La demanda residual del producto *k.* en el escenario en donde se encontró la infactibilidad.

4.4.3.4 Ordenamiento de los Escenarios (GRASP^{A} solamente).

Es de esperarse que el costo de una solución del GRASP^{A} dependa en buena medida del orden en que se toman en cuenta los escenarios, y aunque el sentido común nos indicaría acomodar en primer lugar escenarios con mayores demandas, en la práctica esta técnica no siempre resulta adecuada, ya que en algunas instancias el algoritmo se quedaba atrapado en soluciones malas.

Para evitar esta situación, se decidió ordenar los escenarios de formas diversas conforme el algoritmo del GRASP^{A} iteraba. Los resultados de este método fueron muy buenos, como podrá observarse en el capítulo de resultados computacionales.

4.5 Ordenamiento de los Productos.

Las soluciones que genera cualquiera de las versiones analizadas del GRASP dependen de manera directa del orden en que se consideran los productos. Esto se debe a que son los primeros productos quienes toman las “mejores” rutas definidas para ellos, saturando las capacidades de las aristas en estos caminos. Para los últimos productos analizados, muchas de las aristas en sus rutas ya fueron saturadas por los demás productos analizados previamente y sólo pueden usar las rutas que no usen tales aristas.

Con el fin de escapar de óptimos locales y de generar soluciones diversas, los productos se ordenan de varias formas durante la generación de soluciones. Esta consideración permite obtener soluciones muy distintas entre sí.

4.6 Fase de Post-Procesamiento

En este punto se analizará la forma en que se mejoran a optimalidad local las soluciones encontradas por la fase constructiva en cualquiera de las tres versiones de GRASP desarrolladas.

El post-procesamiento consiste en un chequeo sistemático a través de las aristas definidas como necesarias en el diseño obtenido, para encontrar las que son posibles eliminar sin comprometer factibilidad en ningún escenario.

Para lograr lo anterior trabajaremos con el problema de ruteo multiproducto que se define en cada escenario (una vez que está fijo cierto diseño), el cual se resuelve de forma muy eficiente con ayuda de CPLEX.

El algoritmo a seguir es el siguiente:

1. Enumerar todas las aristas contenidas en el diseño encontrado por GRASP (cualquier versión).
2. Sistemáticamente recorrer una a una las aristas y eliminarlas del diseño.
3. Correr cada escenario del problema original como un problema individual de ruteo multiproducto y comprobar factibilidad.
    a. De encontrar factibilidad en todos los escenarios, eliminar la arista en cuestión de manera definitiva, siempre que el hacerlo represente una reducción en costo del diseño y distribución de los productos.
    b. Si se encuentra infactibilidad en algún escenario, no continuar el chequeo por escenarios, simplemente añadir de regreso la arista analizada y regresar al punto 2 para seleccionar otra arista.

Conforme se eliminan aristas de un diseño entregado por GRASP podemos observar que, de manera general la suma total de los Costos Fijos disminuye (ya que se elimina una arista) y la suma de los Costos Variables aumenta. Si nuestro problema es

del tipo de Costos Fijos relevantes, el eliminar una arista nos beneficiará siempre, ya que el ahorro en Costos Fijos absorbe el costo extra en los Variables.

De manera opuesta ocurrirá en problemas de Costos Variables relevantes, en donde eliminar una arista y ahorrar en Costos Fijos, puede llevarnos a un gasto mayor por concepto de aumento en la suma de los Costos Variables.

CAPÍTULO 5. DISEÑO DE EXPERIMENTOS

5.1 Introducción.

A fin de realizar la experimentación de la metodología propuesta, fue necesaria la creación de instancias con características lo más cercanas a la realidad. En este capítulo, describiremos la forma en que fueron construidas las instancias que sirven como validación y comparación de los métodos propuestos en esta tesis.

Así mismo, otros tópicos importantes que se tratarán en este capítulo son: la generación de un archivo manipulable que identifique a nuestros problemas bajo aleatoriedad y la creación de archivos con información del problema (útil para nuestros programas ejecutables), así como la selección y experimentación con los parámetros de implementación.

5.2 Parámetros de las metodologías GRASP.

Para las tres versiones de GRASP diseñadas en la presente obra, se determinaron ciertos parámetros bajo los cuales se corrieron los programas, mostrando buen desempeño y efectiva ejecución.

Los parámetros de relevancia son:

- Número de iteraciones GRASP (*grasp_iter*) predeterminadas para encontrar una buena solución al problema.
- Valor de la l o factor de peso para la función de evaluación *E(s*, ruta p)*
- Incremento y Decremento del Valor de ? (?$^{+}$ y ?$^{-}$)
- Umbral de Sustitución en el conjunto Élite *(h)*
- Umbral de Diversidad de la Población *(e)*
- Número de rutas cortas iniciales

- Cardinalidad de LRC

En las gráficas 1 a 3 se muestran los promedios de los costos de las soluciones de todas las instancias probadas en esta tesis, bajo diversas combinaciones de parámetros a lo largo de los ajustes:

- Valor de la l: 1, 10
- Incremento del Valor de ?: 2, 5, 25
- Decremento del Valor de ?: 2, 5, 25
- Umbral de Sustitución en el conjunto Élite *(h)*: 0.10, 0.30, 0.60
- Umbral de Diversidad de la Población *(e)*: 0.10, 0.30, 0.35, 0.60
- Cardinalidad de LRC: 10%, 20%, 30%, 60%

Para todas las gráficas de este capítulo, tendremos que el eje "x" exhibe los parámetros seleccionados para una corrida en particular (el orden es ?, ?+, ?-,η, e, %LRC); el eje "y" nos muestra las desviaciones promedio o máxima según se indique, del costo de las soluciones para todas las instancias del grupo especificado, con respecto al costo de una solución encontrada por Scatter Search para cada escenario de datos, esto es, los puntos graficados para cada combinación de parámetros son el promedio de las desviaciones de todas las instancias de prueba para la máxima desviación (según el caso) respecto a una cota superior de cada escenario. Las tablas con los resultados graficados en este capítulo pueden consultarse en los Anexos (Tablas A1 – A3).

5.2.1 Número de Iteraciones GRASP.

Este número refleja la cantidad de veces que le es permitido al algoritmo iterar para encontrar una buena solución. Es de esperarse que entre mayor sea este valor, las soluciones sean mejores, aunque el tiempo de cómputo se verá incrementado en función directa a este aumento de iteraciones.

Para todas las versiones de GRASP se decidió permitir 1,000 iteraciones en general, y adicionalmente, para el $GRASP^A$ se hicieron pruebas con 10,000 iteraciones para observar el comportamiento de la metodología en cuanto a mejoras en la calidad de

soluciones. Por lo general, si bien existían mejoras en algunos puntos porcentuales, el tiempo de cómputo se elevaba a 10 veces comparado con tan solo 1,000 iteraciones.

5.2.2 Valor de *l* o Factor de Peso para la Función de Evaluación *E(s*, ruta p).*

Recordemos que a fin de obtener conocimiento de soluciones previamente generadas se utiliza la función *E(s*, ruta p),* la cual combina la calidad de una selección con la frecuencia que aparece esa selección en buenas selecciones generadas previamente. El parámetro "*?*" guiará el GRASP hacia la generación de soluciones buenas relativas a la función objetivo (función de valor), o bien hacia soluciones con elementos característicos fuertemente determinados y consistentes (función de intensidad). Este parámetro *?* requiere de un valor inicial, así como de un valor de incremento, y uno de decremento.

Algunos autores que han utilizado funciones similares en otros problemas sugieren iniciar *?* con un valor de 100 (Fleurent y Glover, 1999 [25]). Los valores de *?* grandes tienden a hacer la población muy diversa, mientras que los valores pequeños de *?* tienden a hacer una población de soluciones similares entre sí. Para nuestras pruebas de parámetros tomaremos los valores de *?*=1 y *?*=10, para observar su comportamiento y seleccionar la de mejor rendimiento para nuestros problemas.

Se decidió considerar un valor de 10, ya que en aplicaciones similares (De Alba, 2004 [16]), este valor había mostrado entregar una diversidad de población suficiente, permitiendo a su vez reducir rápidamente este valor a valores cercanos a 1 los cuales proporcionan una población muy similar entre sus soluciones miembro. Para nuestra gráfica 1, el eje "x" muestra el valor de *?*, aumento y reducción del mismo respectivamente, y para el eje "y" las desviaciones promedio y máxima según corresponda. El valor de 1 para *?*, entregó mejores resultados en cuanto a la calidad de soluciones, como puede verse en la gráfica 1. Definitivamente la diversidad de las soluciones será menor, pero se decidió sacrificar diversidad en busca de calidad.

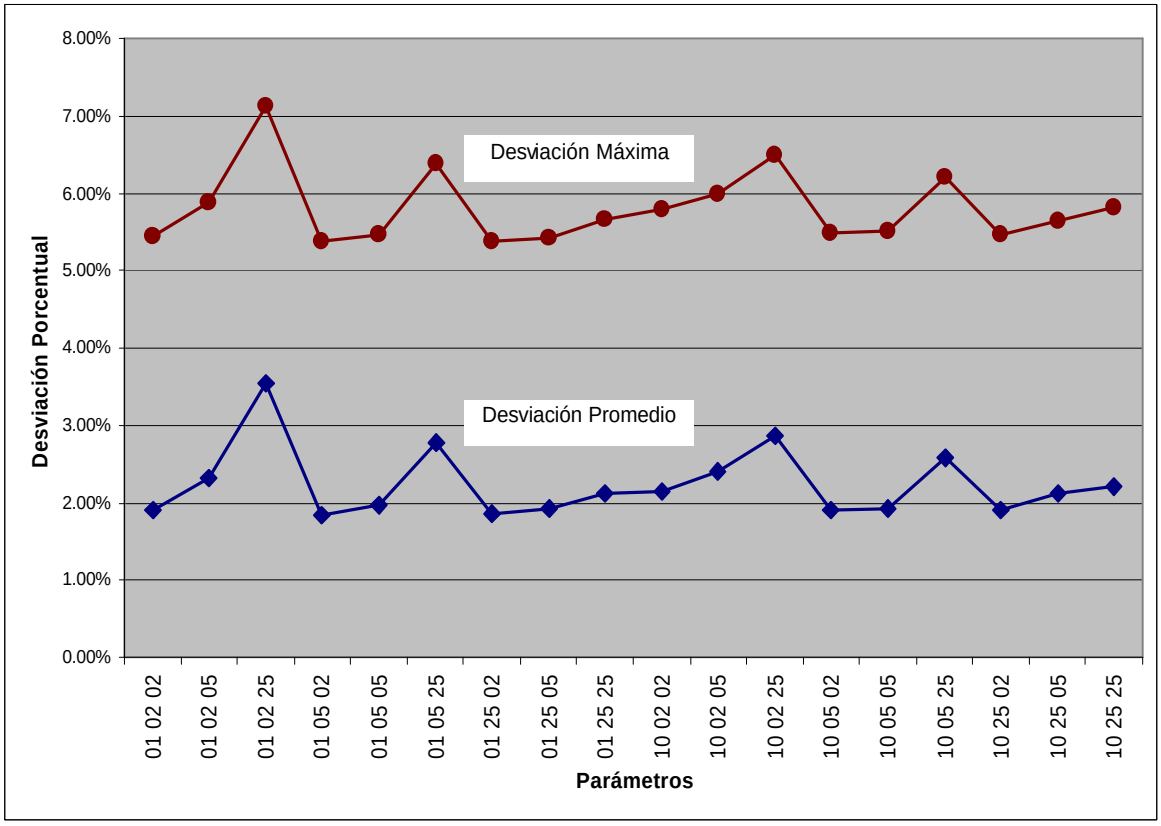
5.2.3 Incremento y Decremento del Valor de *?*.

Partiendo del valor inicial de *?*, a medida que la población de soluciones que se ha ido creando se considera diversa, este valor es decrementado, para dar mayor peso a la calidad en las soluciones que se crearán en las siguientes iteraciones. Al calcular la diversidad de la población y llegar a ser más baja que el umbral pre-establecido de diversidad de la población, este valor se incrementaba nuevamente.

Se hicieron pruebas con diversos valores considerando el valor de incremento y decremento iguales, o bien diferentes. Los mejores resultados mostraron ser los que decrementaban rápidamente el valor de *?* pero lo aumentaban gradualmente.

En la gráfica 1 puede observarse que el mejor rendimiento se encontró con una *?* =1, pero para *?* =10, no hay decrementos considerables en la calidad de soluciones. Un aumento de 5 y reducción de 2 para el valor de *?* fueron los mejores. Puede observarse también que hay una buena cantidad de puntos que ofrecen relativamente buenas soluciones, sin embargo conforme se van fijando más y más parámetros se hace mayor la diferencia en calidad de las soluciones. El resto de los parámetros de implementación no incluidos en este momento para búsqueda fueron fijados en una cantidad arbitraria (*?*=0.30, *e*=0.30, %LRC=0.10).

La siguiente gráfica 1 muestran las desviaciones promedio del conjunto total de instancias de prueba para cada conjunto de parámetros, así como las desviaciones máximas entre todos los escenarios de cada una de las instancias. Las tablas con los datos de esta gráfica se encuentran en el apartado de Anexos (Tabla A1).



Gráfica 1. Ajuste de ?, ?+ y ? -. Desviaciones Promedio y Máxima respecto a mejor solución encontrada por Scatter Search para cada escenario de datos.

5.2.4 Umbral de Sustitución en el Conjunto Élite (*?*) .

Este parámetro permite decidir la membresía de las soluciones recién creadas en el conjunto élite de soluciones buenas y distantes. Para lograr acceder a este conjunto una solución deberá tener mejor calidad que la mejor solución en el conjunto élite, o bien, ser mejor que la peor solución en el élite y además ser suficientemente distante (es decir, no "tan" parecida) a las que ya integran este conjunto. En este caso, el término "suficientemente distante" implica que la distancia de la solución a todas las soluciones que integran el conjunto élite deberá ser mayor que *?*.

El valor de "*?*" deberá encontrarse entre los valores 0 y 1. Si se tiene un alto valor de *?* (cercano a uno), entonces para que una nueva solución pueda ingresar al conjunto élite, deberá ser no sólo buena respecto a su función objetivo, sino además muy diferente del

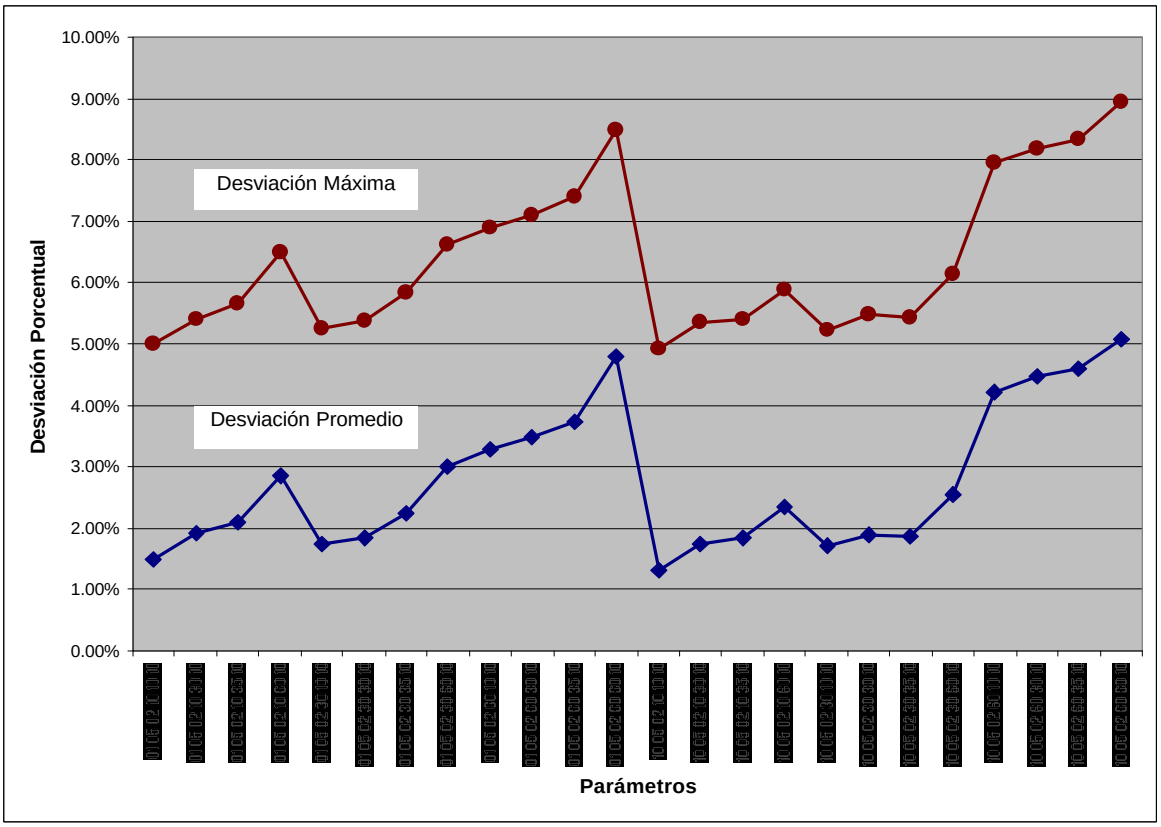
resto de las soluciones que integran el conjunto élite. Esto haría que muy pocas soluciones sustituyeran las del conjunto élite, debido a que las buenas soluciones normalmente son hasta cierto grado “parecidas”. Un valor muy bajo de *?* implicaría no tomar en cuanta el factor diversidad en el procedimiento.

Las mejores soluciones son encontradas para un valor bajo del umbral de sustitución en conjunto élite (*?*=0.10), es decir, permitir a soluciones parecidas o no “tan diversas” acceder al conjunto élite. Valores de *?*=0.30 y *?*=0.35 fueron también relativamente buenos.

Para nuestra gráfica 2, el eje “x” muestra el valor de *?*, aumento y reducción del mismo, umbral de sustitución, diversidad de población y porcentaje de LRC respectivamente; para el eje “y” muestran las desviaciones promedio y máxima según corresponda (la desviación máxima mostrada es el promedio de desviaciones máximas encontradas por instancia).

5.2.5 Umbral de Diversidad de la Población (*e*).

Este parámetro *e* se utilizará para guiar el incremento o decremento del parámetro *?* en la función *E(s*, ruta p)* en dependencia de la diversidad de la población de soluciones creadas. Esta diversidad se calcula como el promedio de las distancias entre cada dos soluciones de la población. Una vez fijos *?* así como sus aumentos y reducciones, se probaron los umbrales de sustitución en el conjunto élite y el umbral de diversidad de la población. En la gráfica 2 se muestran los resultados encontrados en la búsqueda de la combinación de parámetros que mejores resultados entregue. Para los parámetros no determinados aún, se tomaron valores arbitrarios fijos. Los mejores resultados fueron obtenidos de nuevo, para valores bajos de *e* (*e*=0.10, *e*=0.30, *e*=0.35).



Gráfica 2. Ajuste de umbral de sustitución y diversidad. Desviaciones Promedio y Máxima respecto a mejor solución encontrada por Scatter Search para cada escenario de datos.

5.2.6 Número de Rutas Cortas Iniciales.

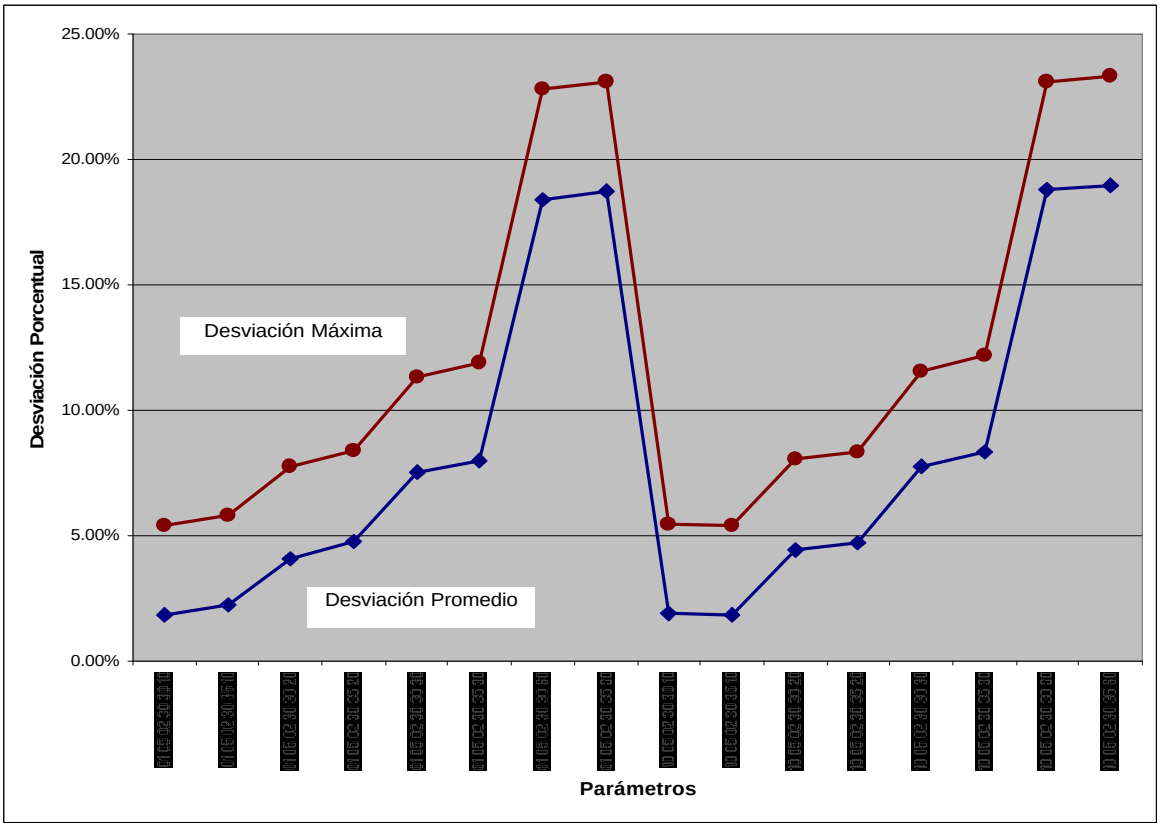
En la versión determinista de nuestro problema (De Alba, Álvarez y González-Velarde, 2003 [17]) se mostró que para un GRASP con características de memoria, característica propia de las tres versiones que aplicamos aquí, a medida que crece el valor de q (número de rutas encontradas para cada producto), mejoran las soluciones del procedimiento, pero empeora notablemente el tiempo de ejecución. Se decidió tomar un valor de q de 60 rutas, que en la práctica obtiene resultados buenos sin deterioro considerable en la ejecución.

5.2.7 Cardinalidad de LRC.

La cardinalidad de la Lista Restringida de Candidatos (LRC) se consideró como una proporción fija de las rutas más cortas determinadas para cada producto, siendo definida como el 10% de las rutas encontradas. Esta cardinalidad mostró dar buenos resultados en la ejecución.

En la gráfica 3 pueden verse los detalles de las pruebas ejecutadas, una cardinalidad de la Lista Restringida de Candidatos de 0.10 entrega los mejores resultados. En el apartado de Anexos (Tabla A3) pueden verse los datos origen de la siguiente gráfica. Los resultados mostrados aquí son el promedio de las desviaciones de todas las instancias de prueba, así como las desviaciones máximas respecto a los escenarios de cada instancia.

Para nuestra gráfica 3, el eje “x” muestra el valor de *?*, aumento y reducción del mismo, umbral de sustitución, diversidad de población y porcentaje de LRC respectivamente; para el eje “y” muestran las desviaciones promedio y máxima según corresponda (la desviación máxima mostrada es el promedio de desviaciones máximas encontradas por instancia).



Gráfica 3. Ajuste de longitud de la LRC. Desviaciones Promedio y Máxima respecto a mejor solución encontrada por Scatter Search para cada escenario de datos.

5.3 Criterio de Comparación.

La validación de los resultados de las metodologías GRASP que fueron implementadas será respaldada por la comparación con trabajos previos, cotas superiores e incluso contra soluciones óptimas, según el caso lo permita.

En la mayoría de los casos, comparar resultados contra el óptimo global de un problema de optimización combinatoria es en particular difícil, debido a la complejidad de este tipo de problemas, que crece exponencialmente respecto al tamaño de la instancia a tratar. Una alternativa en esos casos es comparar contra la mejor solución entera encontrada por un optimizador comercial en un determinado tiempo específico, lo cual representaría una cota superior. Al incluir escenarios nuestro problema se vuelve virtualmente imposible de tratar de forma exacta, a menos que lo desglosemos por escenarios y resolvamos un problema determinista para cada uno. Esta información

puede darnos idea de qué tan buena es nuestra solución robusta, como veremos más adelante. Sin embargo para redes grandes, estos problemas también son difíciles de resolver de forma exacta.

La optimización estocástica puede dar cotas inferiores a nuestro problema, sin embargo es muy complicado desarrollar una metodología completa para tener cotas, y el tiempo de nuestra investigación es limitado.

Para valorar la calidad de las soluciones obtenidas, procederá como sigue: la solución (diseño) que entrega cada una de nuestras versiones GRASP será implementada en cada uno de los escenarios posibles y se calculará el costo total (esto es, costo del diseño más costo de transportación en ese escenario). Lo ideal sería comparar este costo total provocado por nuestra solución en cada escenario con el costo de la solución óptima del problema de diseño determinista en cada escenario pero, como ya se mencionó, esto también es un problema retador aún para instancias de tamaño mediano. Muchas veces luego de 6 horas, el CPLEX no ha encontrado siquiera una solución factible al problema determinístico en un escenario. Es por ello que el costo total de nuestra solución en cada escenario será comparado contra el costo de una buena solución factible de este escenario (cota superior), la cual se obtiene usando la metodología de Búsqueda Dispersa (De Alba, 2004 [16]).

Otra comparación de costo de solución y tiempo computacional será contra un trabajo que analiza un problema como el nuestro (Medina, 2005 [61]), pero con una heurística de búsqueda que toma algunas ideas de la metodología de Búsqueda Tabú.

En conclusión, el criterio de evaluación para medir el desempeño de nuestro algoritmo será el de comparación contra cotas superiores, así como contra trabajos previos. Cabe destacar que las cotas superiores que nos servirán de comparación y que poseen *buena calidad*, en muchas ocasiones exhibieron mejor valor que el encontrado por el optimizador comercial CPLEX en un tiempo predeterminado de 8 hrs.

5.4 Generador de Instancias.

Para ser capaces de probar la validez de nuestros procedimientos, fue necesario un generador aleatorio de instancias de red. Dado que nuestro problema no ha sido muy estudiado, no existen muchas bases de instancias de redes disponibles.

Sin embargo existen las instancias con las cuales se trabajó durante la validación de un trabajo previo (Medina, 2005 [61]), y serán usadas con fines de comparación. Es necesario, sin embargo generar instancias de variadas características, para observar el comportamiento de nuestro método de solución ante las más diversas circunstancias, incluso no reportadas en trabajos anteriores al nuestro.

El generador de instancias, tal como se trabajó con él, crea un archivo con información de una red. Por lo tanto cada instancia de red bajo incertidumbre 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 tipología 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 por este procedimiento será usado como entrada para los diversos programas que se han elaborado. 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 que traten el problema de redes.

Parte de las propiedades del generador de instancias es obtener redes con diferentes grados de capacidad y densidad de red. 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 Instancias de red.

El archivo de una instancia de red tendrá la siguiente información:

- Número de nodos, aristas, productos y escenarios en el registro tipo 1

- Probabilidades por escenario en registros siguientes tipo 2
- Origen – Destino de arista, Costo Fijo y Capacidad de la arista. Registros tipo 3
- Costos Variables por producto por escenario. Registros tipo 4
- Demandas por producto por escenario. Registros tipo 5
- El registro tipo 1 es único, los registros tipo 2 hay tantos como escenarios tiene el problema. Registros tipo 3 son igual al número de aristas y registros tipo 4 existirán en cantidad igual a (número de aristas)*(productos) *(escenarios). De los registros tipo 5 habrán tantos como productos por escenario.

5.4.2 Parámetros del Generador de Instancias.

Con el objetivo de generar nuevas instancias con características similares a las instancias disponibles, se implementaron los mismos parámetros utilizados en el trabajo de Medina (2005 [61]). Esto permitirá realizar la comparación con los resultados de ese trabajo.

A continuación describiremos 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 de la red, se ha predefinido este valor entre 55 y 100.

C Entero que representa el costo variable. En un escenario específico y para cualquier producto, los costos variables tomarán valores 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 de la capacidad posible de la arista. Este valor se usa conjuntamente a LS explicado más abajo. Se predefinió a 0.9

LS Número fraccionario que representa el límite superior del rango de capacidad. Es una fracción de la capacidad posible de la arista. Este valor se usa conjuntamente a LI. Predefinido a 1.1.

A Número entero que representa la constante de capacidad. Este valor puede incrementarse a fin de obtener redes holgadas en capacidad. Predefinido a 1. Este parámetro tiene relación con lo que se explicará en el punto 5.4.5 más adelante (A= ℑ).

5.4.3 Procedimiento General.

El procedimiento general para generar aleatoriamente una instancia de red para el problema DRCM se describe a continuación. Cabe señalar que fue utilizada la función aleatoria uniforme *rand( )* integrada en el lenguaje de programación C para la obtención de números aleatorios.

- Se generan aleatoriamente las parejas origen-destino para cada producto en la red. Estos dos números se toman en forma aleatoria entre 1 y N, el número de nodos en la red. Tanto los nodos origen como los nodos destino se obtienen de tal forma que no existan dos productos con el mismo par origen destino.
- Se genera aleatoriamente la demanda para cada producto como un número entero entre 55 y 100, la máxima demanda posible. Se obtiene la demanda promedio dividiendo la suma de demandas de los productos entre el número de productos.
- 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 en la red, de otra forma, es desechada.

- Para cada arista que formará parte de la red se le obtendrá la capacidad, el costo fijo y los costos variables, estos últimos en dependencia de cada uno de los productos en la red.
    a. La capacidad de la arista es un número aleatorio en el rango [LI, LS] * demanda promedio * A, la constante de capacidad.
    b. Se obtiene el costo variable por producto de la arista como un número aleatorio entre 50 y 100, el máximo costo variable. Se obtiene el costo variable promedio de la arista sumando los costos variables por producto obtenidos y dividiéndolo entre el número de productos.
    c. El costo fijo de la arista es igual al costo variable promedio multiplicado por A, la constante de costo fijo.
    d. Una vez obtenidos los valores anteriores, se procedió al almacenamiento de la instancia en un archivo de texto siguiendo el modelo propuesto con anterioridad.

5.4.4 Instancias Creadas.

Los tipos de instancias que fueron empleados para la validación de nuestro procedimiento (ya sean tomadas de las ya existentes o bien generadas de manera aleatoria con características similares) se muestran a continuación:

Red Tipo	NODOS	PRODUCTOS	DENSIDAD	COSTOS RELEVANTES	CAPACIDAD
1	30	30	50%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
2	30	30	75%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
3	30	50	50%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
4	30	50	75%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
5	50	30	50%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
6	50	30	75%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
7	50	50	50%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada
8	50	50	75%	Costos Fijos	Holgada
				Costos Fijos	Ajustada
				Costos Variables	Holgada
				Costos Variables	Ajustada

Tabla 1**. Características de las Instancias Tratadas**

La primera columna de la Tabla 1, corresponde al tipo de red en dependencia de sus dimensiones, la segunda columna es el número de nodos que posee esta familia de instancias, seguido por el número de productos que poseen, así como las respectivas densidades. Luego se define si la instancia es de Costos Fijos o Variables relevantes y por último si las capacidades de las aristas son holgadas o ajustadas. De cada especialización se generaron 5 instancias, dando un total de 20 instancias por familia y 160 instancias en total.

Los nombres de los problemas serán dados según sus características, comenzando con un identificador para reconocer fácilmente la versión del generador aleatorio de instancias que lo creó. Así, tendremos por ejemplo un archivo “pa303050” para una instancia creada por el generador aleatorio original de problemas (“pa”), que posee 30 nodos, 30 productos y una densidad de 50%.

5.4.5 Razón de la Capacidad de una Arista.

Para determinar el grado en capacidad, tanto de redes holgadas como restringidas, se implementó la siguiente razón de capacidad:

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

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:

$MF^k(s) \geq d^k(s) \; \forall \; k \hat{I} K, \; s \hat{I} S.$

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. Por ello 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.

CAPÍTULO 6. RESULTADOS COMPUTACIONALES

6.1 Introducción.

Una vez descritas las características de las instancias utilizadas en el experimento computacional, ya sean generadas de forma aleatoria y/o instancias empleadas en trabajos previos, es momento de discutir los resultados obtenidos por cada una de las versiones GRASP que fueron implementadas.

Los parámetros utilizados son:

- Número de iteraciones GRASP: 1,000 y 10,000 (para comparación en el $GRASP^A$)
- Valor inicial de la l: 10
- Incremento y Decremento del Valor de ?: 5 y 2 respectivamente
- Umbral de Sustitución en el conjunto Élite *(h)*: 0.30
- Umbral de Diversidad de la Población *(e)*: 0.35
- Número de rutas cortas iniciales: 20 de cada tipo (3 tipos)
- Cardinalidad de LRC: 10% de la cantidad de rutas generadas para cada producto

Como fue explicado en el capítulo 5, el criterio de comparación para la metodología propuesta es respecto a una cota superior de buena calidad, obtenida mediante un procedimiento basado en Búsqueda Dispersa (De Alba, 2004 [16]), y adicionalmente se harán comparaciones (cuando sean posibles) con un trabajo previo que considera incertidumbre (Medina, 2005 [61]).

Las gráficas en este capítulo muestran resultados promedio para los tipos de instancias estudiados. Para el eje “x” se muestran los tipos de instancias y para el eje “y” de las gráficas 4 y 5 tiempos, y para el resto de las mismas (6 a 25) desviaciones promedio respecto a una cota superior de buena calidad.

Los experimentos computacionales fueron realizados simulando los datos inciertos mediante variables aleatorias independientes uniformemente distribuidas. Si bien no se realizó un experimento específico para validar el desempeño del algoritmo bajo otras distribuciones o dependencias estadísticas entre las variables podemos conjeturar que al considerar otras distribuciones o incluir dependencias estadísticas el algoritmo se desempeña en forma similar. Basamos esta conjetura en que el principal pilar de éxito del algoritmo es el aprovechar adecuadamente la estructura matemática del problema, intentando, independientemente del valor de la demanda, encontrar un diseño que sea capaz de satisfacerla al menor costo posible.

6.2 Tiempos de corridas con GRASP.

Las diferentes versiones de GRASP (con Memoria, Robusto y Ajustado) tuvieron corridas por 1,000 iteraciones y para el $GRASP^A$ se corrieron también 10,000 iteraciones de manera adicional, por haber entregado las mejores soluciones y con el objetivo de observar su comportamiento a lo largo de corridas más extensas.

Los tiempos de cómputo fueron muy parecidos para todas las versiones GRASP con la obvia excepción del $GRASP^A$ de 10,000 iteraciones, el cual entregó tiempos de aproximadamente 10 veces los de la versión con 1,000 iteraciones.

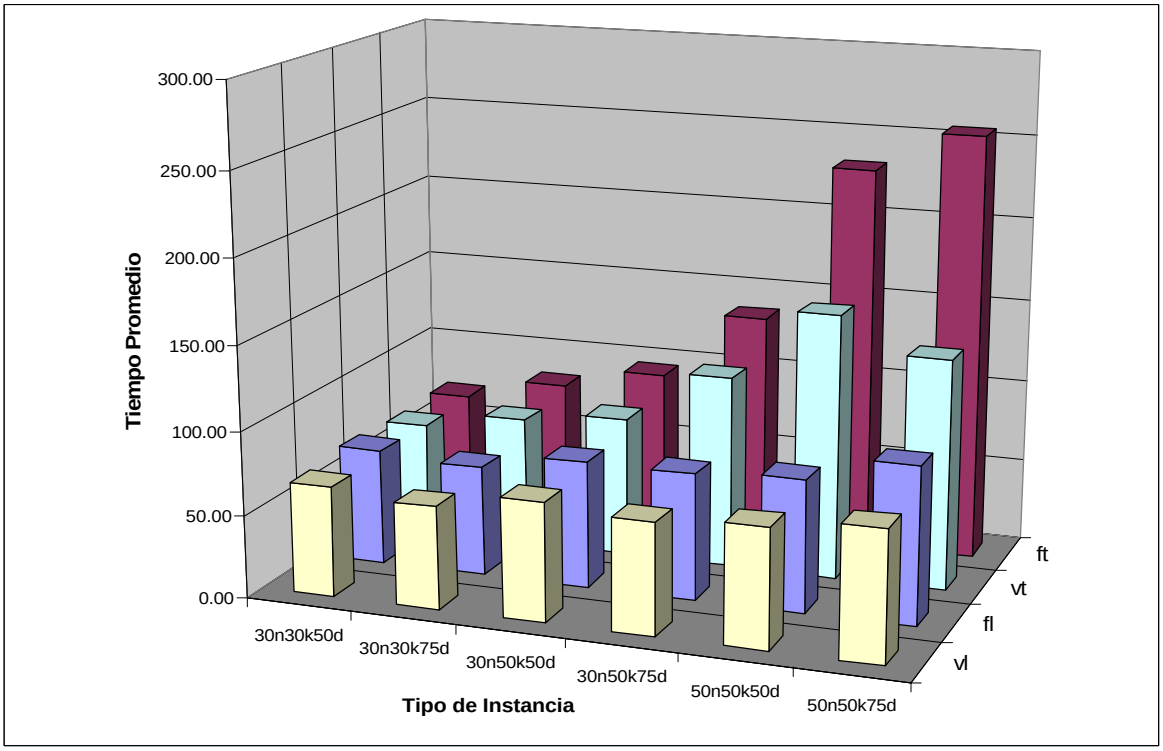
La gráfica de tiempos para el $GRASP^A$ de 1,000 y 10,000 iteraciones se muestra a continuación (gráfica 4). No se incluyen tiempos de otras versiones ya que nuestro algoritmo más representativo y eficiente fue el aquí mencionado, y las versiones GRASP anteriores solo serán puntos de referencia para este. Las tablas se encuentran en los Anexos (A4).

En la gráfica 4 podemos observar que para los problemas de características de red holgada el tiempo de cómputo no aumenta mucho conforme las dimensiones del problema crecen, a diferencia de los problemas de características ajustadas de red, en donde pequeños crecimientos en dimensiones de problema conlleva a grandes crecimientos en el tiempo de cómputo. La razón de este crecimiento aparentemente se encuentra en el hecho de que el algoritmo encuentra una mayor dificultad para estas instancias en encontrar las rutas adecuadas para incorporar a las soluciones. Sin

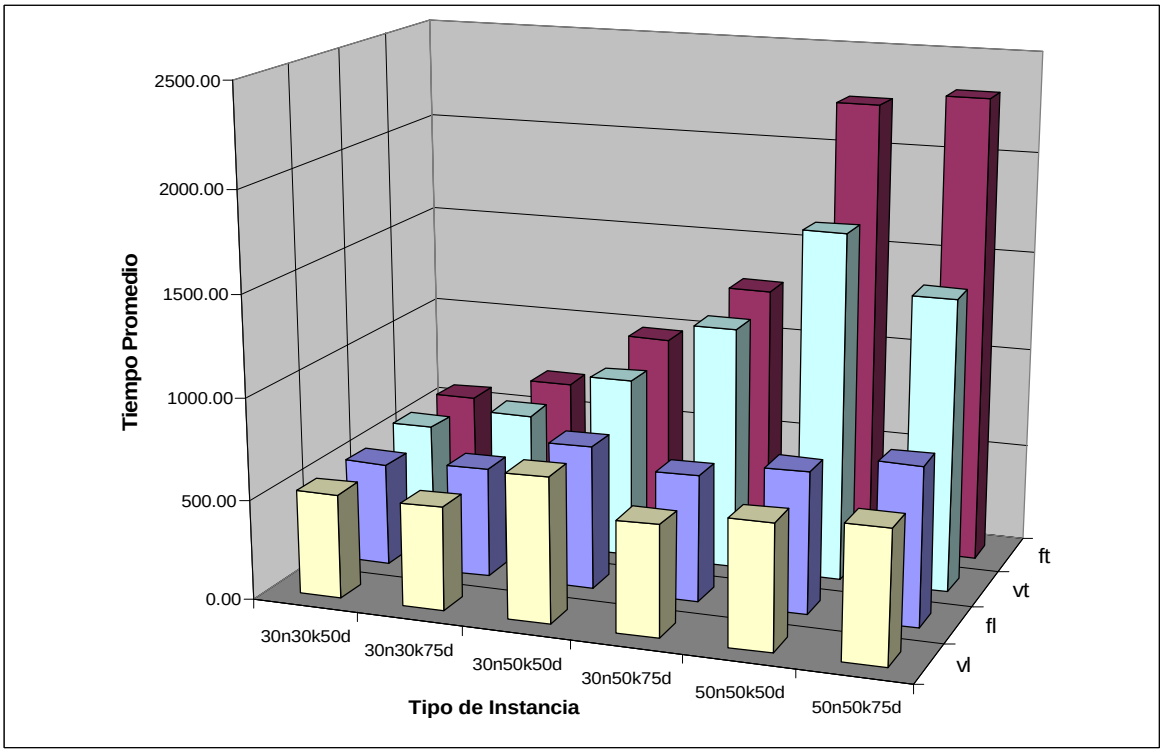
embargo, los tiempos para la mayoría de las instancias es relativamente pequeño, del rango de apenas minutos y en los casos más tardados menos de 250 segundos aproximadamente para 1,000 iteraciones de $GRASP^A$.

En contraparte, tenemos la gráfica de tiempos para 10,000 iteraciones de $GRASP^A$ (gráfica 5), en donde los tiempos se ven aumentados como es de esperarse, pero el crecimiento es lineal, con lo que tenemos que para 10 veces más iteraciones habrá 10 veces más tiempo; y lo que antes fuera un máximo de 250 segundos aproximadamente, se convierte en alrededor de 2,200 segundos (37 minutos aprox.)

Sin embargo este comportamiento es satisfactorio para nuestros fines, ya que sabemos el costo computacional de aumentar iteraciones al algoritmo. Debemos tener presente que aún y cuando se incrementen las iteraciones, las mejoras en la calidad de la solución serán cada vez menores, por el hecho de que a cada momento estamos más y más cerca del óptimo y será incrementalmente más difícil mejorar una solución.



Gráfica 4. Tiempos de ejecución del GRASP^{A} con 1,000 iteraciones.



Gráfica 5. Tiempos de ejecución del GRASP^{A} con 10,000 iteraciones.

6.3 Resultados de Desviaciones Promedio para las Versiones GRASP.

En este punto discutiremos el comportamiento de la calidad de las soluciones encontradas mediante las diversas versiones GRASP. Para ello, la solución que entregue el GRASP será implementada en cada escenario posible de datos y su costo (costo del diseño más costo de operación en ese escenario) será comparado contra una cota superior del costo óptimo del problema de diseño determinista en este escenario. Esta cota, como se ha comentado anteriormente, es de muy buena calidad (De Alba, 2004 [16]).

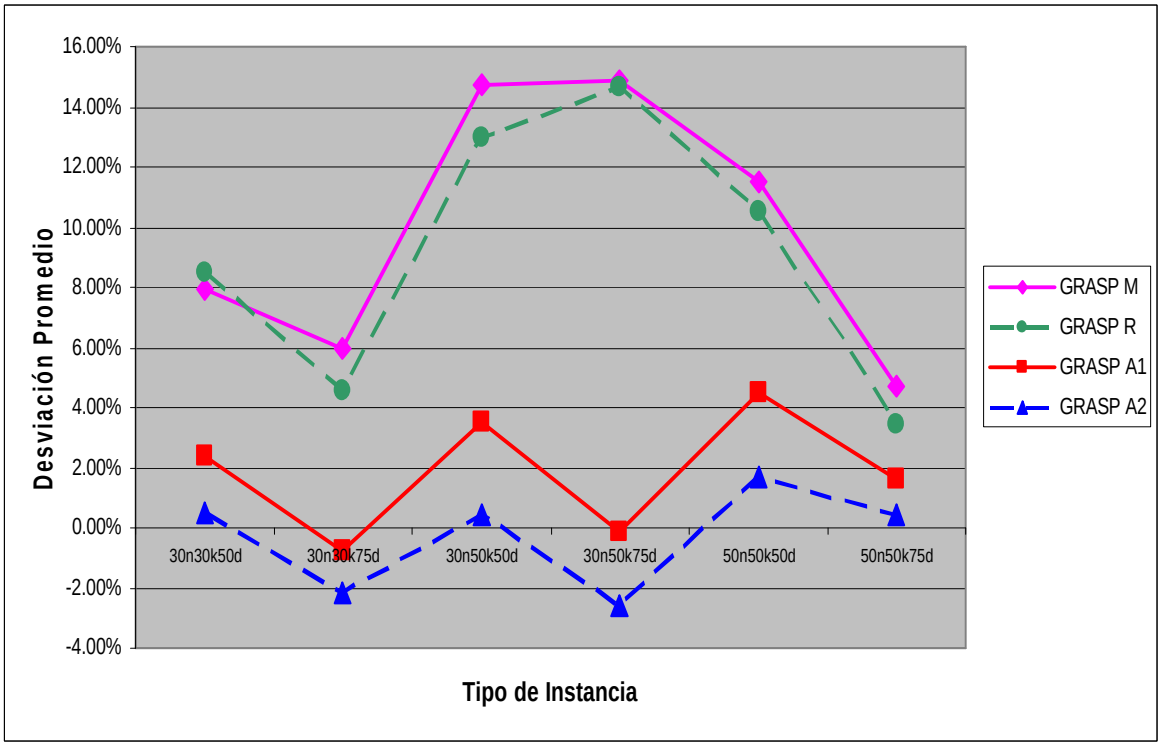
Estas desviaciones serán promediadas sobre todos los escenarios y son los valores que se considerarán en el eje "y" de la gráfica 6. En dicha gráfica podemos observar las desviaciones promedio agrupadas por tipo de instancia, que son aquellas que comparten características de dimensiones (número de nodos, número de productos a transportar y densidad de red). Los nombres de los grupos de instancias nos indican estas características y en ese orden, así un problema tipo "30n30k50d" posee 30 nodos, 30 productos y una densidad de red de 50%. En el eje "y", el 0% representa la cota superior del óptimo del problema determinístico en ese escenario. En esta gráfica solo se presentan los resultados de la fase constructiva de cada versión GRASP, con fines comparativos de los algoritmos.

El $GRASP^M$ se denota como "grasp M" en la gráfica, el Robusto como "grasp R", para el Ajustado de 1,000 iteraciones tenemos "grasp A1" y "grasp A2" para el de 10,000. La tabla con los datos de esta gráfica se encuentra en los Anexos (Tabla A6).

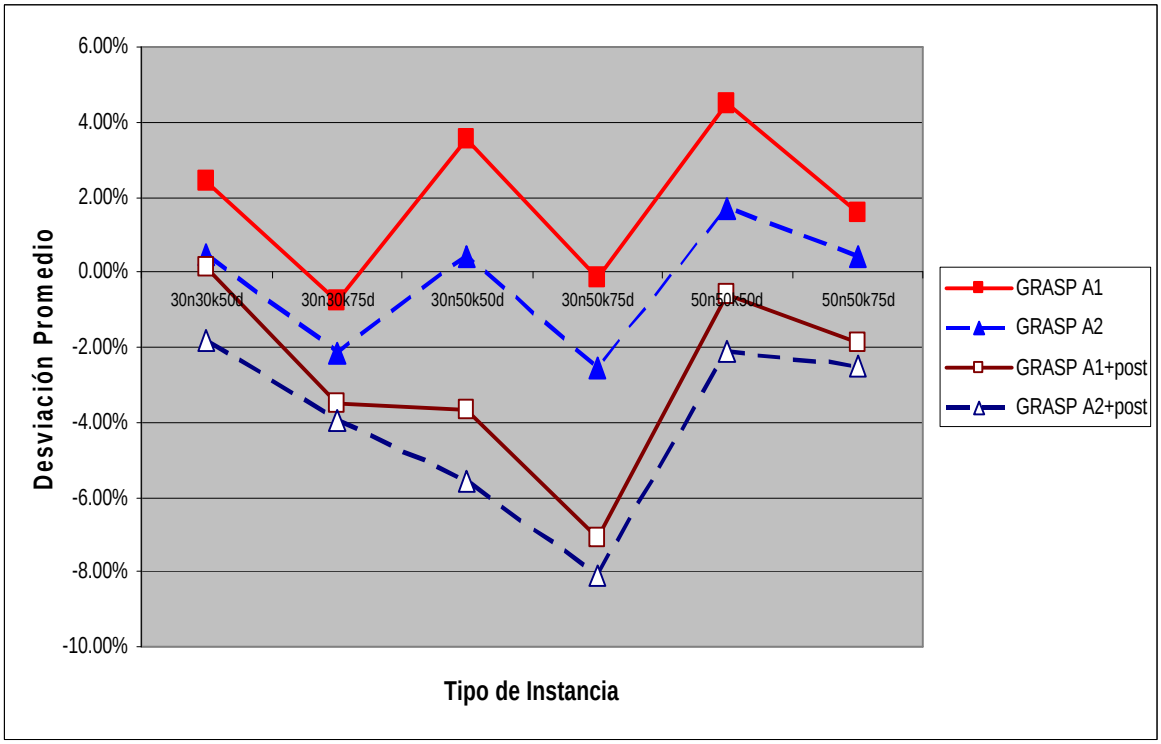
Como podemos observar de la gráfica, las mejores soluciones son por mucho las de la versión $GRASP^A$ de 10,000 iteraciones, seguido muy de cerca por su versión con 1,000 iteraciones. Aquí podemos hacer un balance entre el tiempo que deseamos de ejecución y la calidad de solución deseada. Las versiones de Memoria y Robustas de GRASP tienen calidades similares, con costos superiores al $GRASP^A$.

Es interesante destacar que por regla general, los problemas que poseen mismas características pero mayor densidad (por ejemplo, 30n30k50d vs. 30n30k75d) obtienen mejores soluciones promedio, lo cual puede explicarse por el hecho de que al poseer más aristas en la red, el algoritmo tiene más opciones para trabajar.

La peor desviación promedio para nuestros problemas es menor al 4%. Implementar 10,000 iteraciones disminuye el costo de las soluciones en aproximadamente un 2.15% más! (Con respecto a las cotas superiores cosideradas).



Gráfica 6. Desviación promedio de las soluciones encontradas por cada versión de GRASP. Versiones GRASP antes de post-procesar.



Gráfica 7. Desviación promedio para GRASP^{A} y post-procesamientos.

Ahora observemos el comportamiento más a detalle de las soluciones del GRASP^{A} de 1,000 y 10,000 iteraciones incluyendo el post-procesamiento para esta versión descrito en la sección 4.5, con el objetivo de investigar la mejora de esta fase final sobre las soluciones construidas por el GRASP^{A}. En la gráfica 7 se muestran únicamente las versiones Ajustadas de 1,000 y 10,000 iteraciones y sus respectivos post-procesamientos, los cuales contribuyen en mejoras porcentuales del orden del 2% y hasta 6% en promedio. Las tablas con los datos de esta gráfica pueden encontrarse en la sección de los Anexos (Tabla A6).

Ahora encontramos que la calidad de las soluciones se ha mejorado considerablemente gracias al post-procesamiento, que además de contribuir positivamente en nuestros resultados, el tiempo de cómputo que consume es muy pequeño, del orden de las decenas de segundos. Es importante destacar que aún y cuando las soluciones de GRASP^{A} con 1,000 y 10,000 iteraciones son importantemente distintas entre sí en cuanto a calidad (2.15% de diferencia con respecto a las cotas superiores), al momento de post-procesar se disminuye considerablemente esa diferencia (baja a 1.40%), lo que nos hace reconsiderar el uso de apenas 1,000 iteraciones de GRASP^{A}, cuando el tiempo de cómputo sea una limitante.

6.4 Resultados de Desviaciones Máximas para las Versiones GRASP.

Teniendo en cuenta que cualquiera de los escenarios de datos puede realizarse en el futuro, y solamente uno de ellos se realizará, tiene sentido poner atención no solo al promedio de las desviaciones de la cota superior de cada escenario, sino también a la mayor de estas desviaciones.

Para cada instancia se calculará la desviación máxima (sobre todos los escenarios) y estas desviaciones son promediadas para todas las instancias pertenecientes a un mismo grupo en la gráfica 8. Es necesario recordar que nuestro algoritmo busca una solución común para varios escenarios y es de esperarse encontrar una tal que pueda tener excelentes rendimientos para la mayoría de los escenarios y para uno o dos muy malos con respecto a las soluciones de Scatter Search que buscan una solución considerando tan solo uno de los escenarios. Lo reportado a continuación

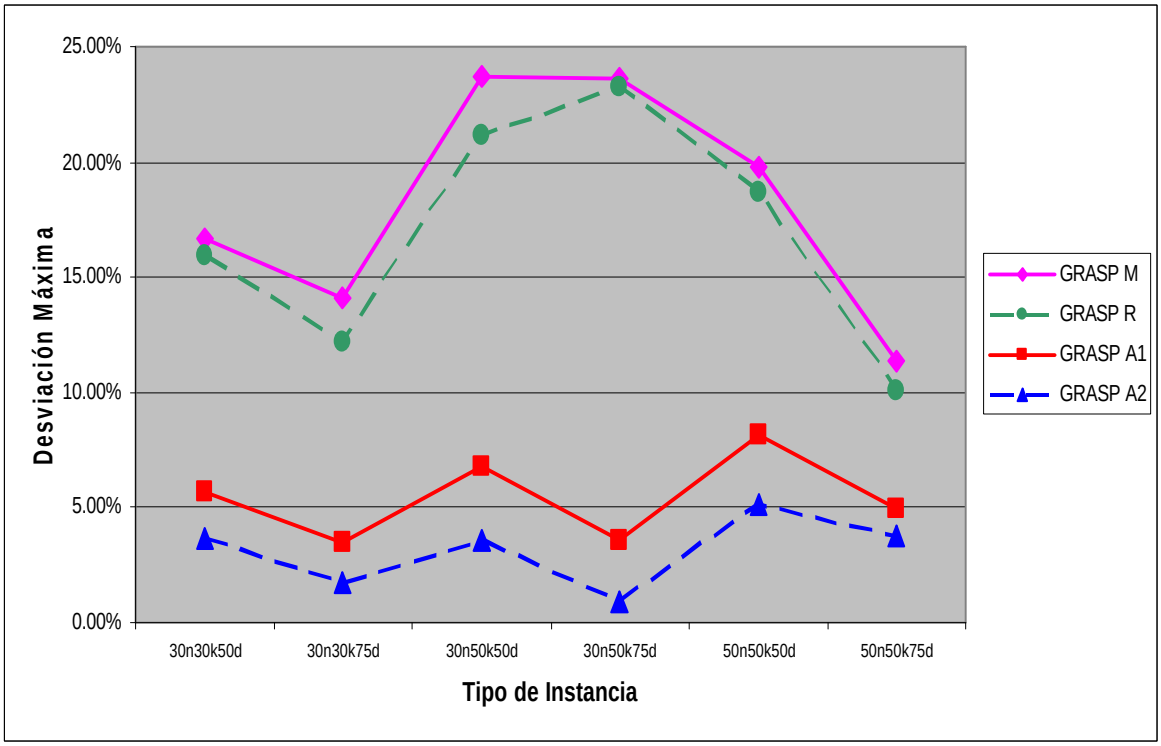
consiste en ese peor rendimiento en alguno de los escenarios de cada problema, promediadas para las instancias de cada tipo.

De nuevo observamos mejores soluciones para las versiones Ajustadas de GRASP comparadas contra las de Memoria y Robustas. Incrementar iteraciones para $GRASP^A$, otra vez entrega mejores resultados como es de esperarse (2.28% en promedio menor). El comportamiento por grupos de instancias es muy constante, en cuanto a mejor rendimiento para los problemas de iguales características pero mayor densidad de red.

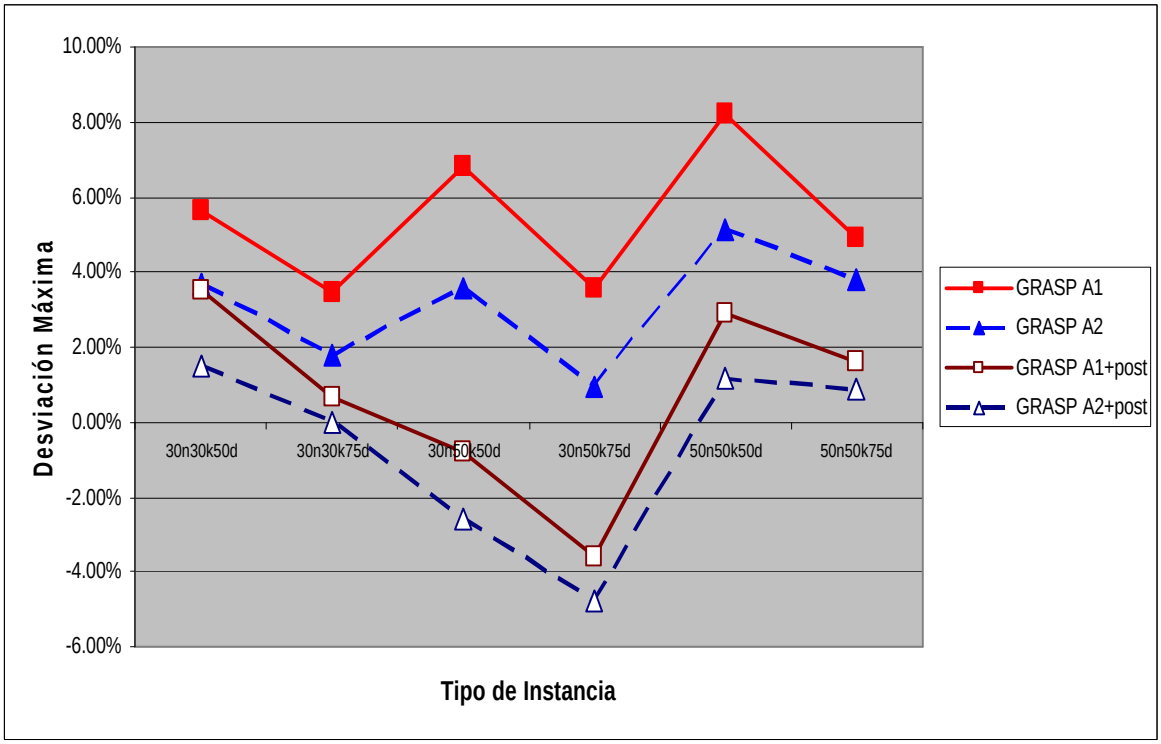
Para las versiones con Memoria y Robustas de GRASP, las desviaciones máximas se encuentran en el rango del 20%, mientras que para los $GRASP^A$ son de 5% para el de 1,000 iteraciones y 3% para el de 10,000.

Es conveniente analizar más a detalle los $GRASP^A$ con post-procesamientos, que logran mejorar las soluciones en alrededor de un 4%-5% con respecto a las soluciones de Scatter Search como se muestra en la gráfica 9 (Datos en tabla de Anexos A7).

Si bien el post-procesamiento logra disminuir la desviación máxima de las soluciones, existen algunas instancias para las cuales la mejora es casi nula, debido posiblemente a que incluso con una mejor solución, alguno de los escenarios es sensiblemente distinto a la mayoría de los demás escenarios y aunque la solución sea muy buena para casi todos los escenarios, queda uno muy mal ajustado.



Gráfica 8. Desviación máxima de las soluciones encontradas por cada versión de GRASP antes de post-procesar.



Gráfica 9. Desviación máxima de soluciones GRASP^{A} y post-procesamientos.

6.5 Resultados de Desviaciones Promedio y Máximas para Problemas tipo FL.

Consideramos que es conveniente analizar a detalle el comportamiento de los algoritmos agrupando las instancias no solamente según sus dimensiones, sino según las características de sus costos y las capacidades de sus aristas.

Analizando las instancias con características de costos fijos relevantes y capacidad de red holgada (FL), tenemos la siguiente información representada en las gráficas de desviación promedio y máxima (gráficas 10 y 11) que se presentan a continuación. Adicionalmente, se incluye una comparación de nuestras diversas versiones GRASP con un trabajo previo (Medina, 2005 [61]) como referencia, los datos que crean estas gráficas se encuentran en los Anexos (Tablas A8).

Podemos observar en las gráficas que nuestras soluciones luego de post-procesamiento son o muy parecidas o mejores que las soluciones del trabajo previo que nos sirve de comparación para el $GRASP^A$ de 1,000 iteraciones con post-procesamiento. En el caso del $GRASP^A$ de 10,000 iteraciones con post-procesamiento, sus soluciones siempre fueron mejores que las del trabajo de referencia.

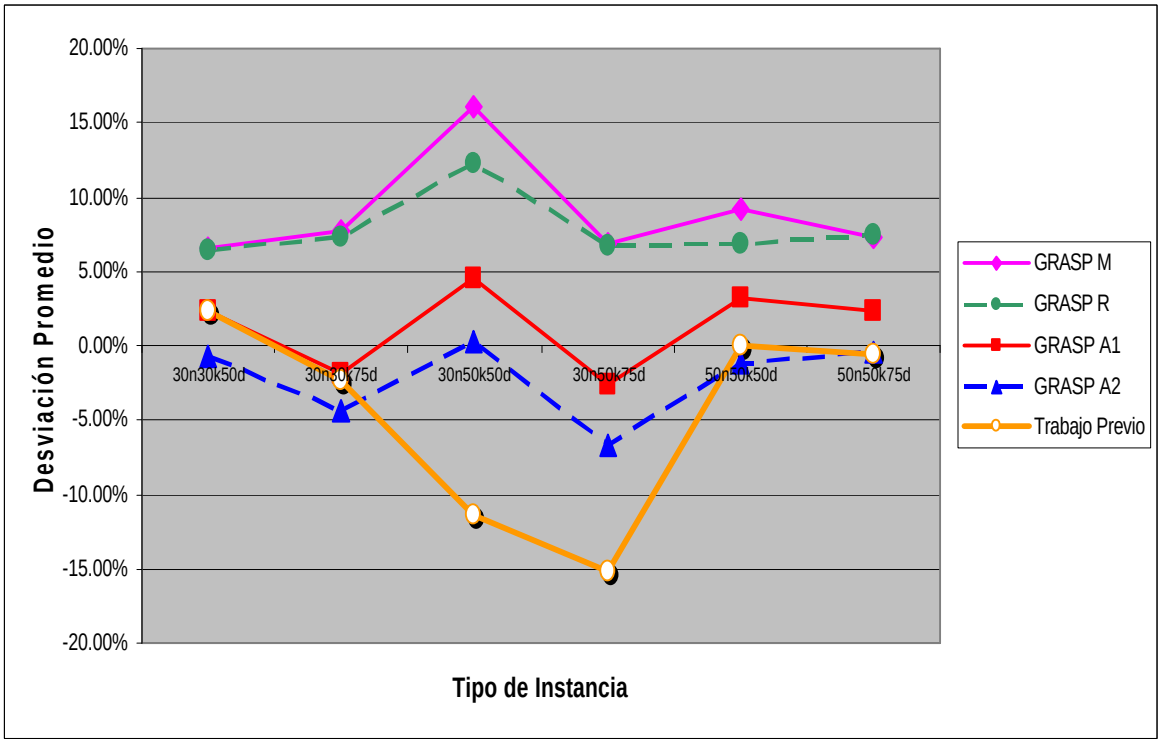
Veamos ahora las gráficas de las desviaciones máximas (gráfica 12 y 13) para comparar nuevamente con el trabajo previo y conocer más a detalle las características de los problemas tipo FL solucionados con nuestras varias versiones GRASP.

Los algoritmos GRASP sin post-procesamiento tienen un rendimiento regular con respecto al trabajo previo, con excepción del $GRASP^A$ de 10,000 iteraciones que muestra buenos resultados.

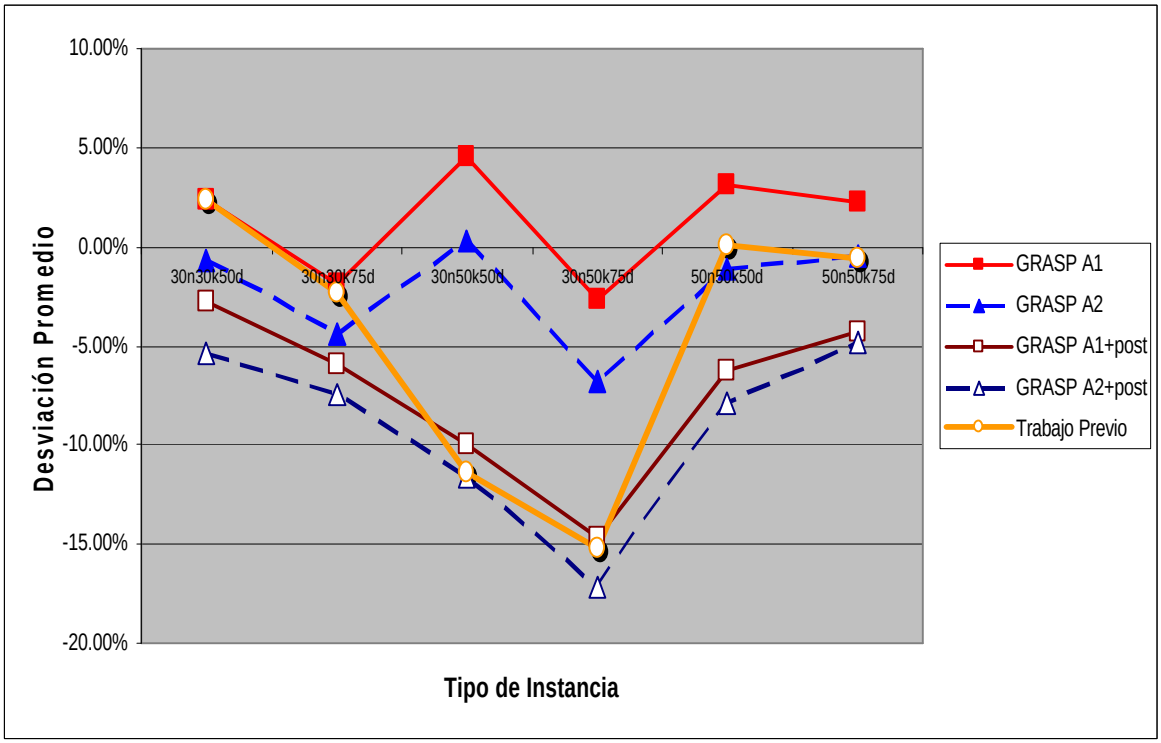
Sin embargo, analizando las versiones que incluyen el post-procesamiento, vemos que ahora se reduce todavía más la distancia entre soluciones del trabajo previo y el $GRASP^A$ de 1,000 iteraciones con post-procesamiento. En cuanto a la versión de 10,000 iteraciones, las soluciones muestran una mucho mejor calidad en este aspecto.

Es necesario aclarar que si bien nuestras soluciones podrían considerarse muy cercanas a las del trabajo previo, nuestra implementación consume un tiempo considerablemente menor, en el rango de los minutos para 1,000 iteraciones o 20-50 minutos para 10,000 iteraciones, mientras que la heurística de trabajo anterior por lo general toma horas de tiempo CPU.

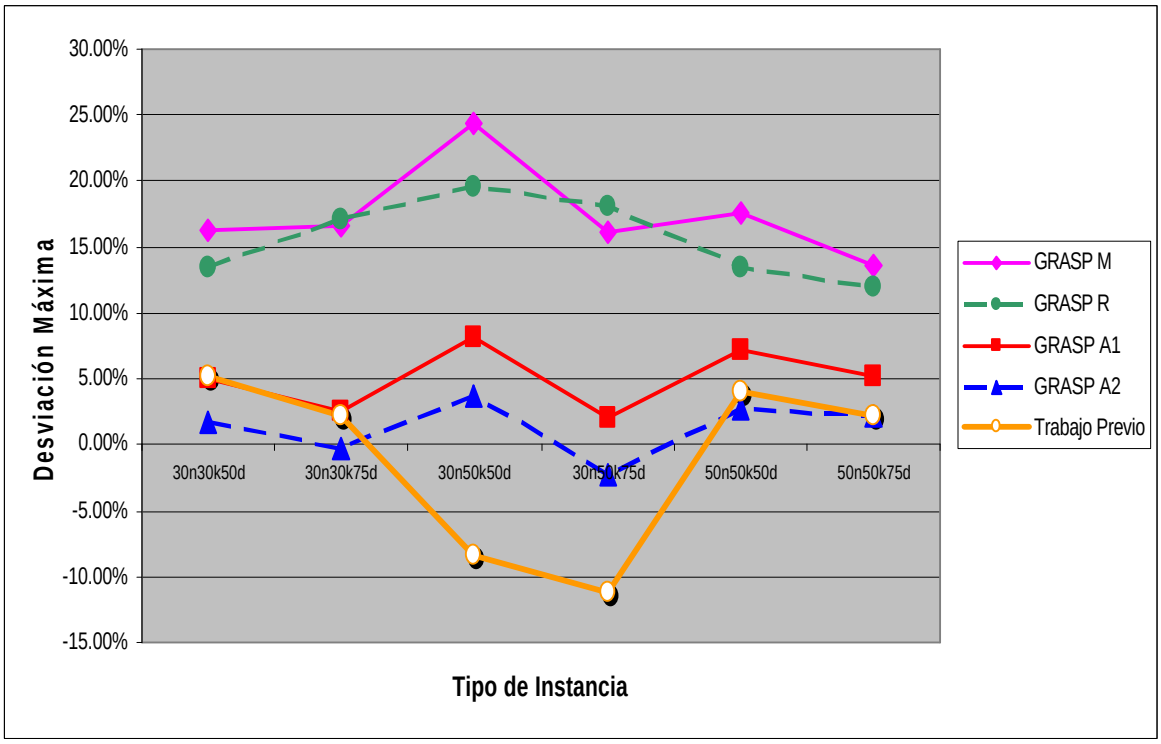
Un posible argumento a favor de la similitud de la calidad de la calidad de las soluciones aportadas por la metodología desarrollada en el presente trabajo y la propuesta por Medina (2005 [61]), es que dichas soluciones están cercanas al óptimo y es difícil obtener grandes márgenes de mejoría. Sin embargo, esto es solo una hipótesis que requeriría ser evaluada con estudios posteriores.



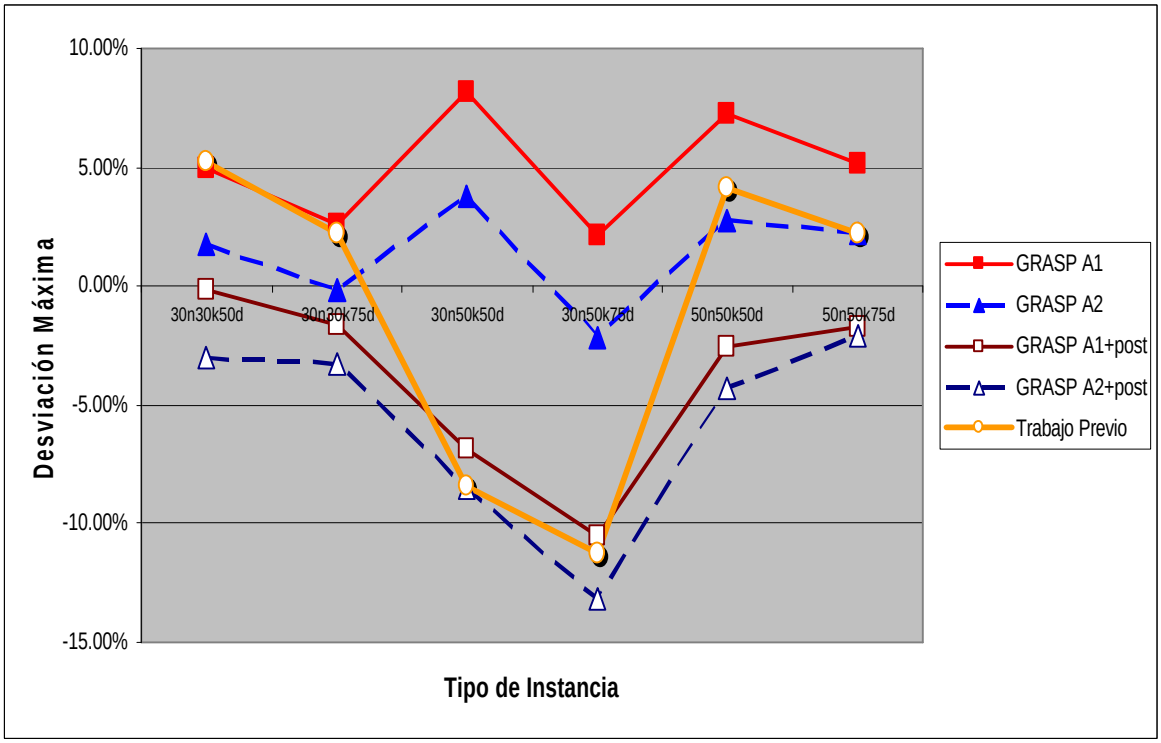
Gráfica 10. Desviación promedio de las instancias FL. Versiones GRASP antes de post-procesar.



Gráfica 11. Desviación promedio FL GRASP^{A} y post-procesamientos.



Gráfica 12. Desviación máxima FL para cada versión de GRASP. Versiones GRASP antes de post-procesar.



Gráfica 13. Desviación máxima FL $GRASP^A$ y post-procesamientos.

6.6 Resultados de Desviaciones Promedio y Máximas para Problemas tipo FT.

En este punto se analizarán las instancias con características de costos fijos relevantes y capacidad justa (FT). Estos problemas son por regla general más difíciles que los de capacidad de red holgada, ya que se hacen necesarias una buena cantidad de rutas para enviar los productos de cada tipo, incrementando dificultad y tiempo de cálculo.

Las gráficas mostradas a continuación (gráficas 14 y 15) comparan de nuevo la calidad en soluciones de varias versiones GRASP y de trabajo previo, así como los resultados incluyendo post-procesamiento de las soluciones de $GRASP^{A}$.

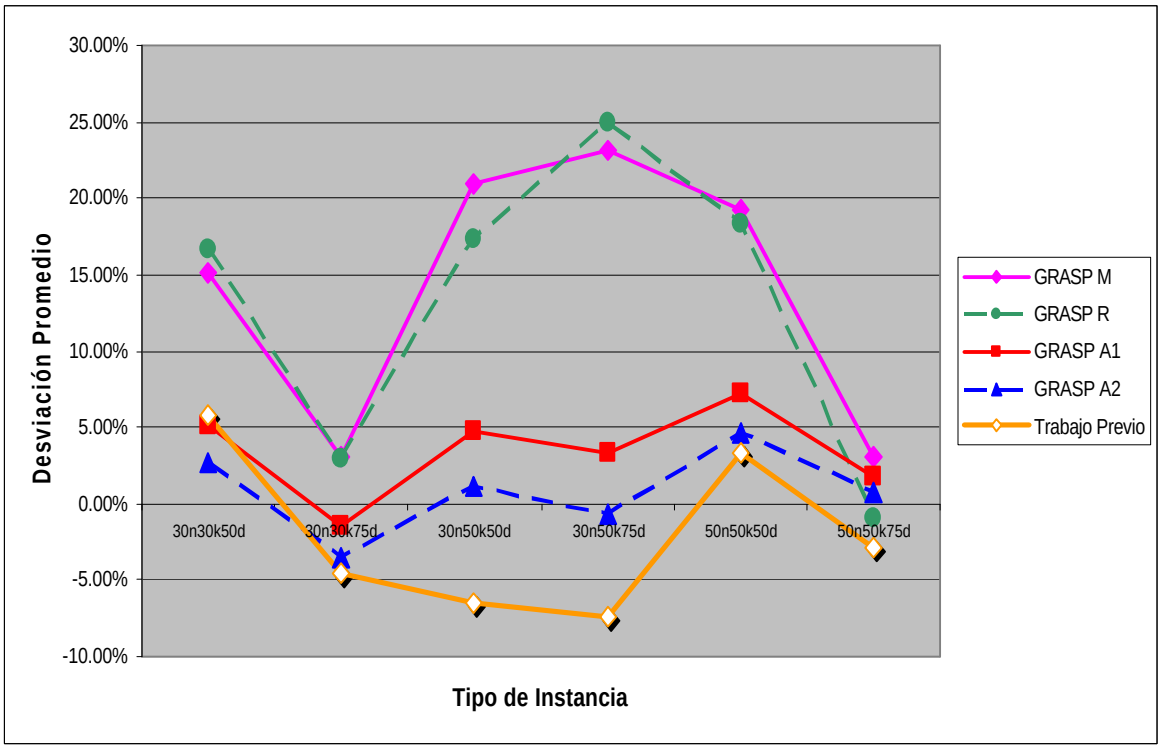
Podemos observar en la gráfica 14 que las soluciones de los $GRASP^{M}$ y Robustos se encuentran por encima de las desviaciones de otros algoritmos, el $GRASP^{A}$ de 10,000 iteraciones sin post-procesamiento está cerca de las desviaciones promedio del trabajo previo, pero cuando consideramos post-procesamiento (gráfica 15) está por debajo siempre. Para 1,000 iteraciones está o muy cercano o por debajo.

Para las desviaciones máximas (gráficas 16 y 17), se obtienen resultados similares. El algoritmo $GRASP^{A}$ de 1,000 iteraciones muestra buen desempeño, comparable al trabajo anterior y el de 10,000 mejor en todos los casos de desviación promedio y máxima.

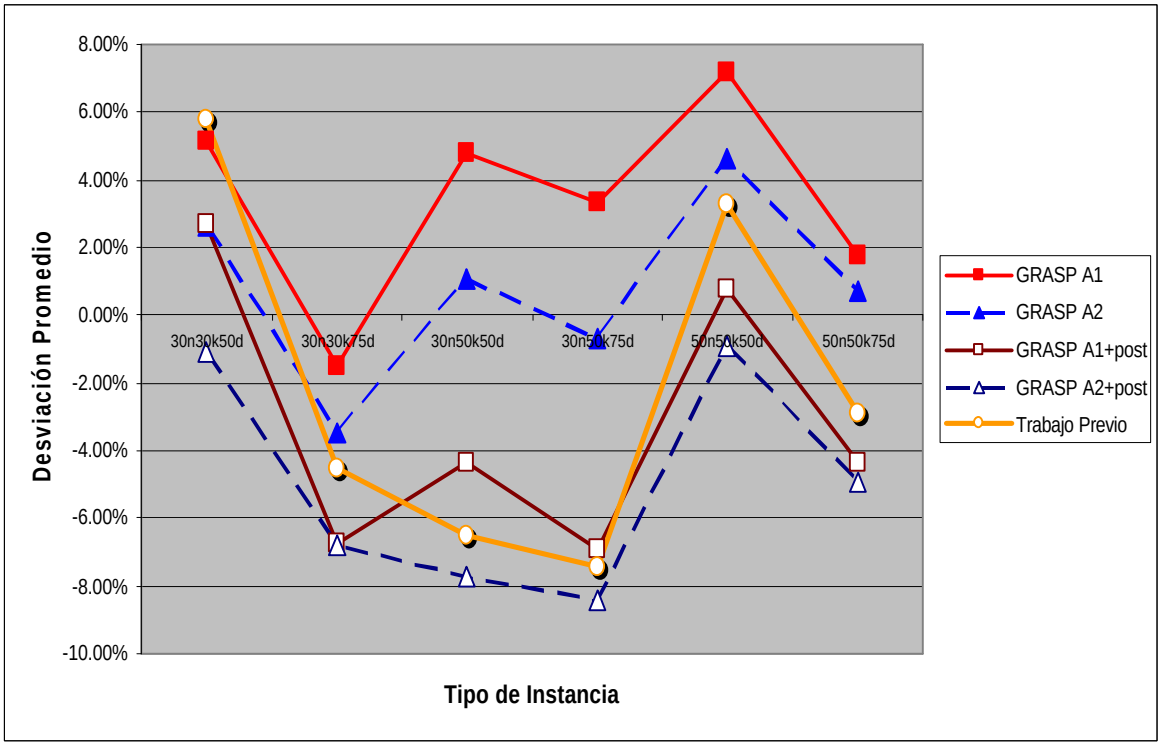
Todos los resultados promedios luego de post-procesar se encuentran por debajo del 0% (es decir, mejores soluciones que las encontradas por Scatter Search para el problema determinista en cada escenario).

Las tablas con los datos que generan estas gráficas pueden localizarse como siempre en la sección de Anexos, bajo el apartado A11.

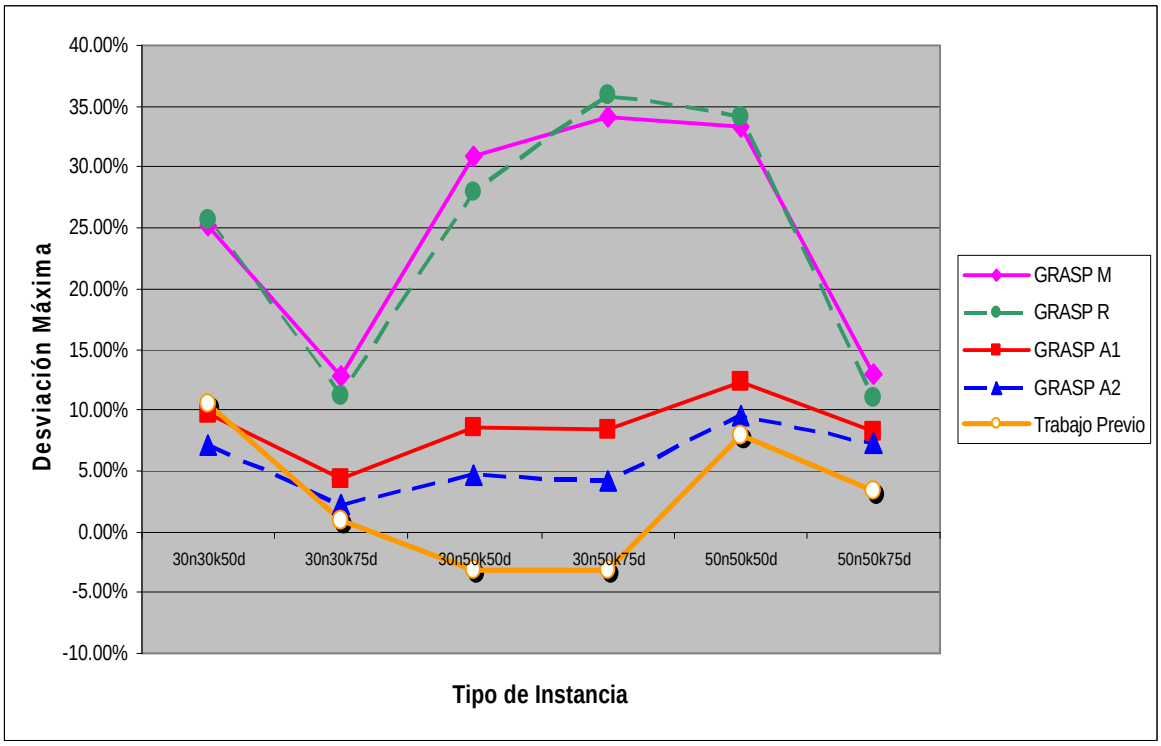
El rendimiento de nuestro procedimiento resulta ser en conclusión, mejor que el del trabajo previo, tanto considerando desviación promedio como para la desviación máxima para los problemas de características FL y FT.



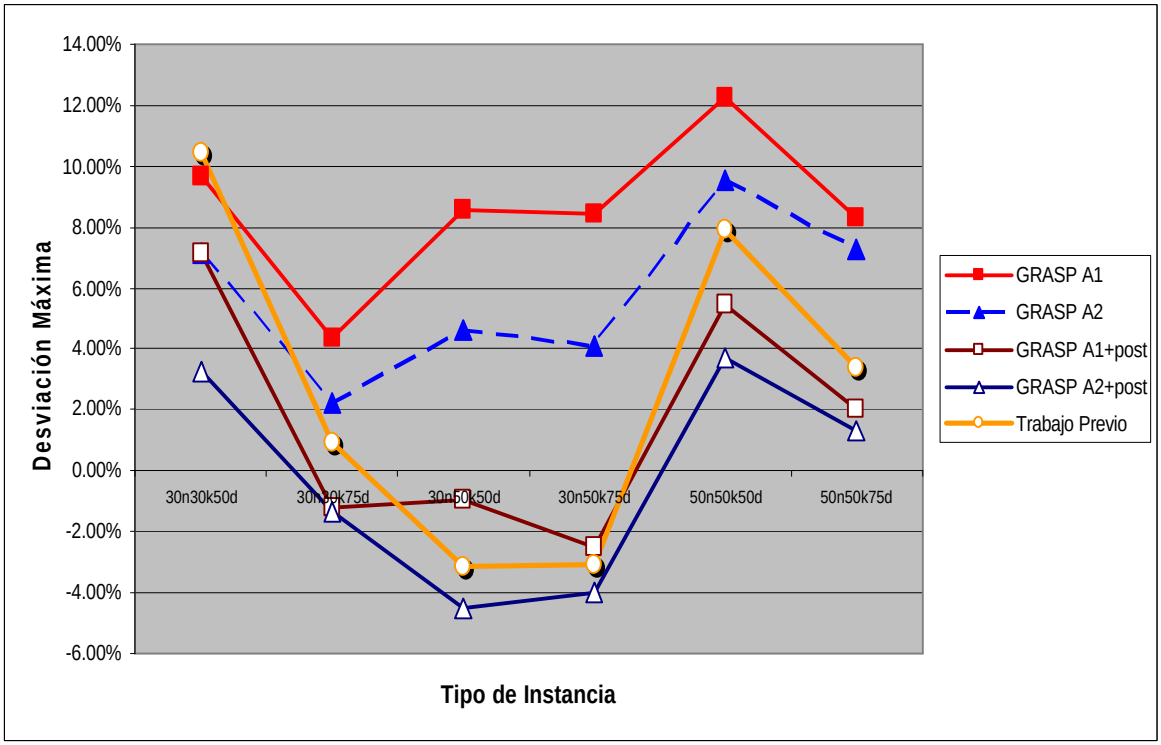
Gráfica 14. Desviación promedio de las instancias FT. Versiones GRASP antes de post-procesar.



Gráfica 15. Desviación promedio FT GRASP^{A} y post-procesamientos.



Gráfica 16. Desviación máxima FT para cada versión de GRASP. Versiones GRASP antes de post-procesar.



Gráfica 17. Desviación máxima FT GRASP^{A} y post-procesamientos.

6.7 Resultados de Desviaciones Promedio y Máximas para Problemas tipo VL.

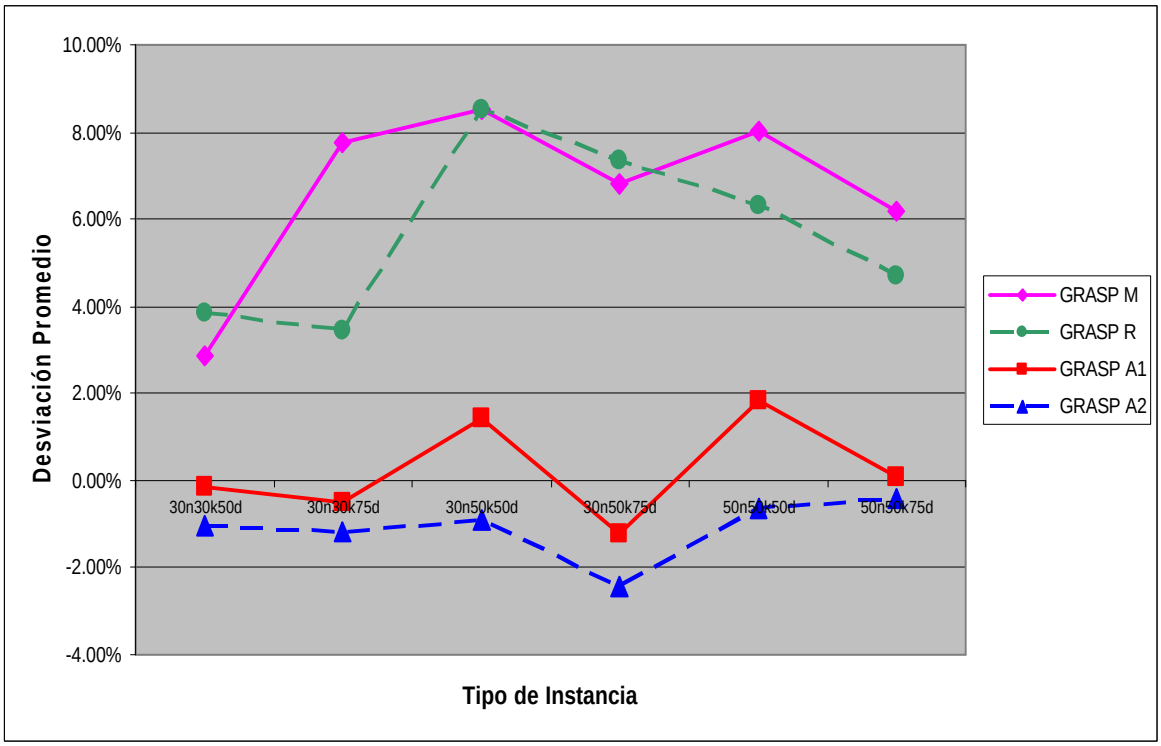
Ahora analizaremos los resultados obtenidos para instancias con costos variables relevantes y capacidades de red holgadas. Estos resultados ya no pudieron ser comparados con el trabajo previo, ya que este último no trabaja con instancias con características de costos variables relevantes, una desventaja comparado contra nuestro $GRASP^{A}$.

En primer lugar tenemos los resultados de desviaciones promedio por tipo de problema, así como el análisis más a detalle de los $GRASP^{A}$ de 1,000 y 10,000 iteraciones con post-procesamiento. Veamos las gráficas 18 y 19.

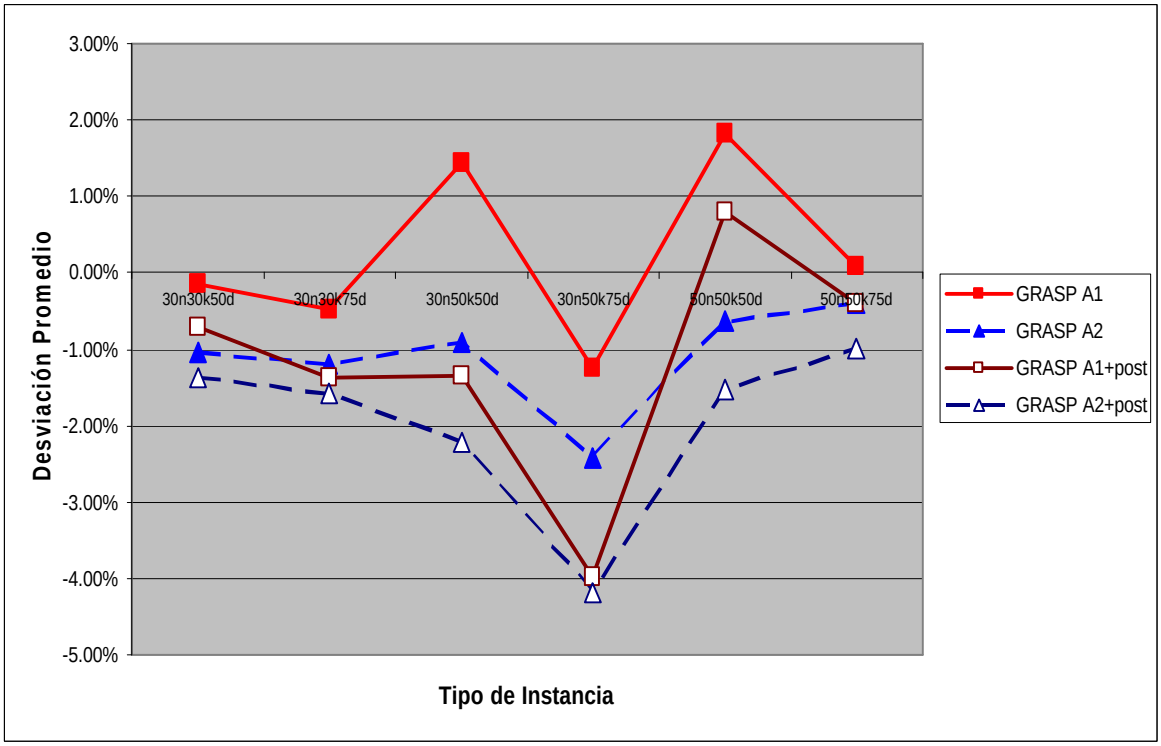
Para estos problemas podemos observar que las versiones de $GRASP^{M}$ y Robustos tienen rendimientos parecidos, aunque relativamente malos. El $GRASP^{A}$ de 1,000 iteraciones tiene buen rendimiento, y el de 10,000 mejora considerablemente las ya de por sí buenas soluciones, terminando con desviaciones promedio por debajo de las soluciones de Scatter Search.

Incluyendo post-procesamiento, las soluciones mejoran todavía más. Si bien se encuentran en un rango de aproximadamente 0.5% ó 1% la mejora es considerable por el hecho de que ya desde antes eran muy buenas soluciones. El algoritmo $GRASP^{A}$ tiene muy buen rendimiento para este tipo de instancias.

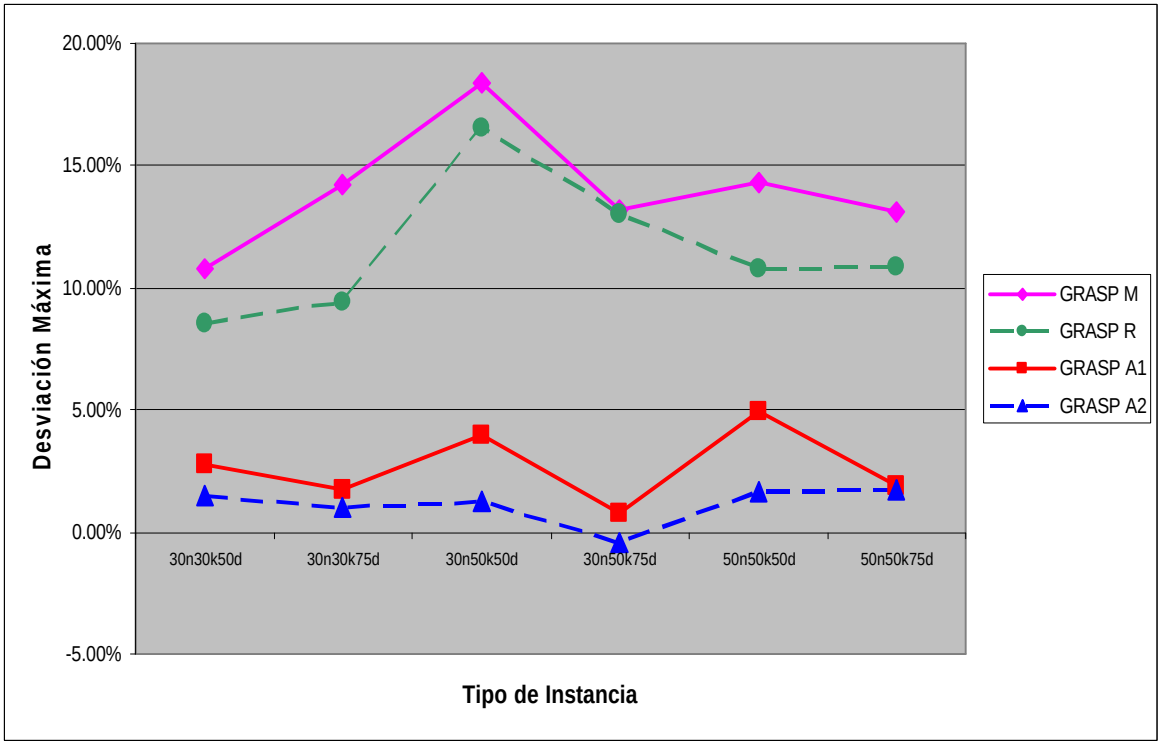
En cuanto a desviaciones máximas tenemos las gráficas 20 y 21. Para estas instancias se pudo observar un fenómeno interesante: al tener soluciones muy buenas, presumiblemente cercanas a las óptimas, las desviaciones máximas incluso luego de post-procesamiento casi no disminuyeron, e incluso en contadas ocasiones empeoraron, ya que en una búsqueda por mejorar la desviación promedio, puede darse el caso de empeorar alguna solución por escenario previamente buena. Al tener soluciones muy buenas un pequeño decremento en desviación promedio, puede ocacionar un incremento en la peor desviación de algún escenario, y verse reflejado en un aumento relativamente drástico en los gráficos, como puede observarse en las instancias tipo 30n30k50d en la gráfica 21.



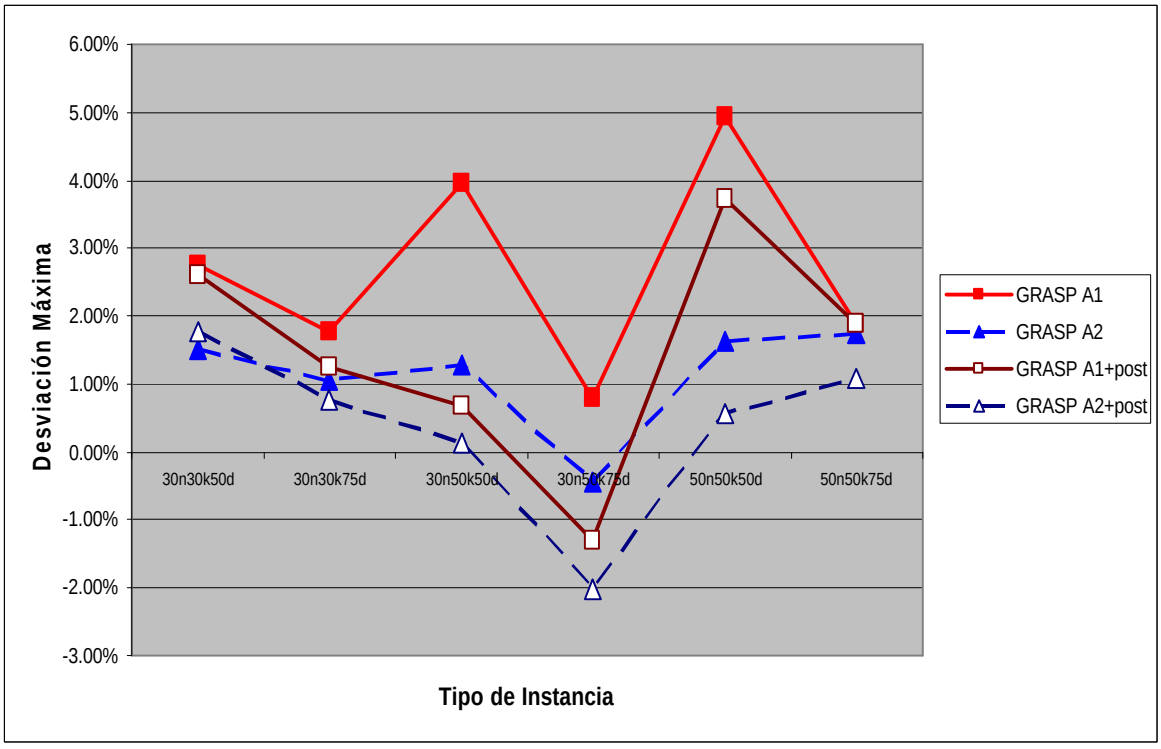
Gráfica 18. Desviación promedio de las instancias VL. Versiones GRASP antes de post-procesar.



Gráfica 19. Desviación promedio VL GRASP^{A} y post-procesamientos.



Gráfica 20. Desviación máxima VL para cada versión de GRASP. Versiones GRASP antes de post-procesar.



Gráfica 21. Desviación máxima VL GRASP^{A} y post-procesamientos.

6.8 Resultados de Desviaciones Promedio y Máximas para Problemas tipo VT.

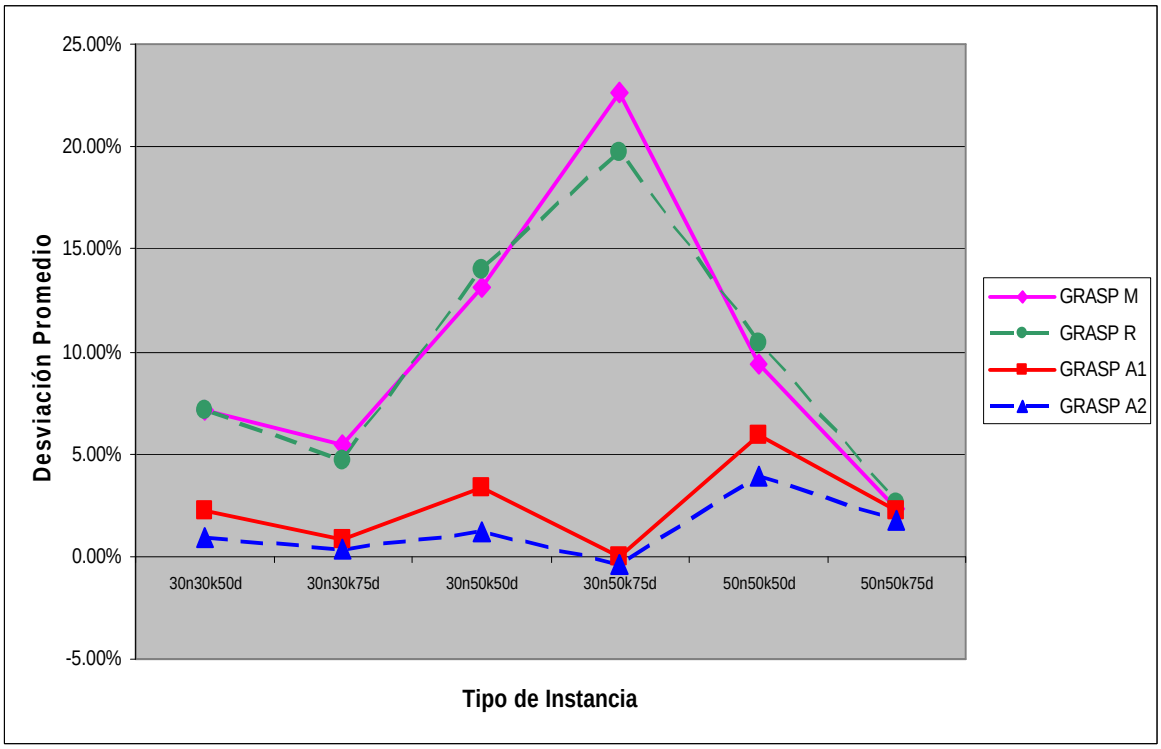
Por último analizaremos las instancias con costos variables relevantes y capacidad ajustada, las cuales son más difíciles de resolver que las de capacidad de red holgada por lo antes mencionado. Las tablas con los datos que generan las siguientes gráficas pueden observarse en los apartados de Anexos A14.

Veamos primero las desviaciones promedio de versiones GRASP (gráfica 22), el análisis más a fondo de post-procesamiento (gráfica 23) y por último los gráficos relativos a las desviaciones máximas (gráficas 24 y 25).

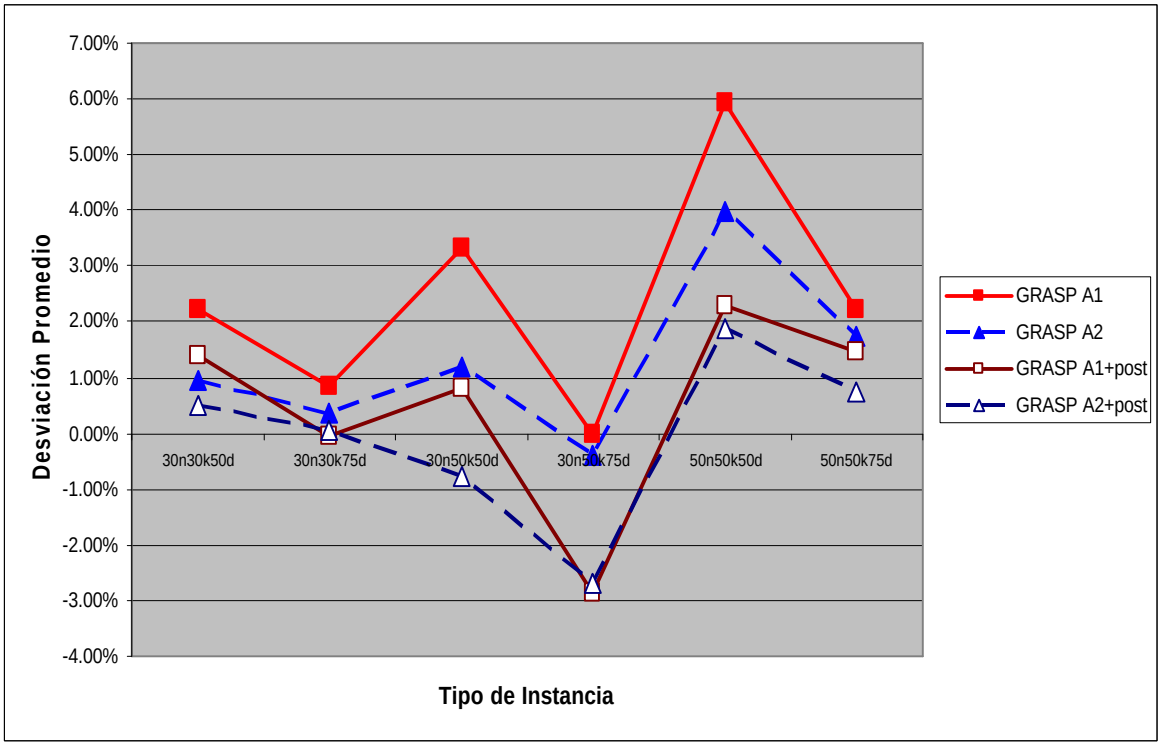
Los resultados para todas las versiones GRASP fueron muy parecidos, en especial para el $GRASP^A$ de 1,000 con el de 10,000 iteraciones, y se dramatiza esta situación al observarlos con post-procesamiento, donde no se encontró mucha mejora. Esto nos hace suponer que las soluciones encontradas desde el algoritmo sin post-procesamiento eran ya muy buenas, como puede verse en las gráficas, con desviaciones de apenas 2% aproximadamente sobre las de Scatter Search.

Para las desviaciones máximas se muestran las gráficas 24 y 25 con resultados. Las desviaciones máximas para las instancias evaluadas con $GRASP^A$ son de apenas 5% por lo general. Incluido el post-procesamiento apenas baja un poco la desviación máxima, pero sigue siendo relativamente buena. Las desviaciones máximas comparadas del $GRASP^A$ de 1,000 y 10,000 iteraciones es prácticamente la misma, sin importar que para 10,000 iteraciones el tiempo de cómputo es 10 veces mayor.

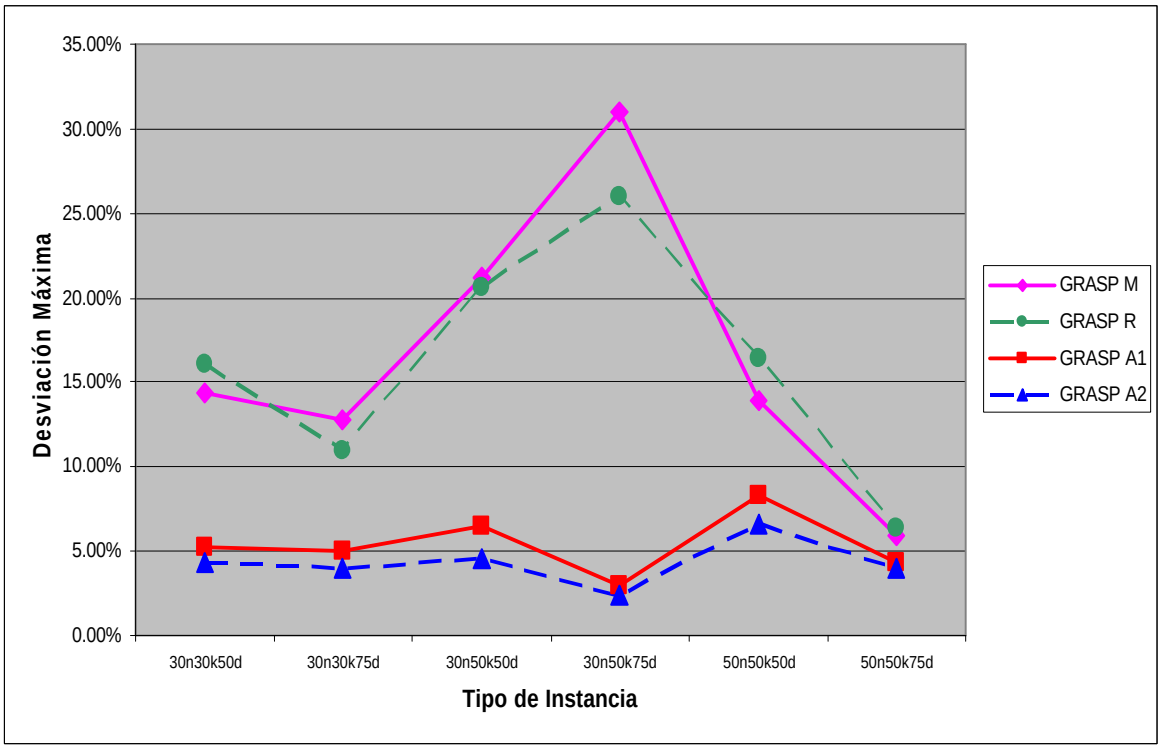
En conclusión, los resultados de nuestro algoritmo de $GRASP^A$ son más que satisfactorios y prometedores, al haber registrado soluciones robustas muy buenas en apenas una fracción del tiempo utilizado en trabajos previos, y sin restricciones en cuanto a tipo de problemas que pueda resolver. Las desviaciones promedio son comparativamente buenas y las desviaciones máximas no comprometen la eficiencia del algoritmo.



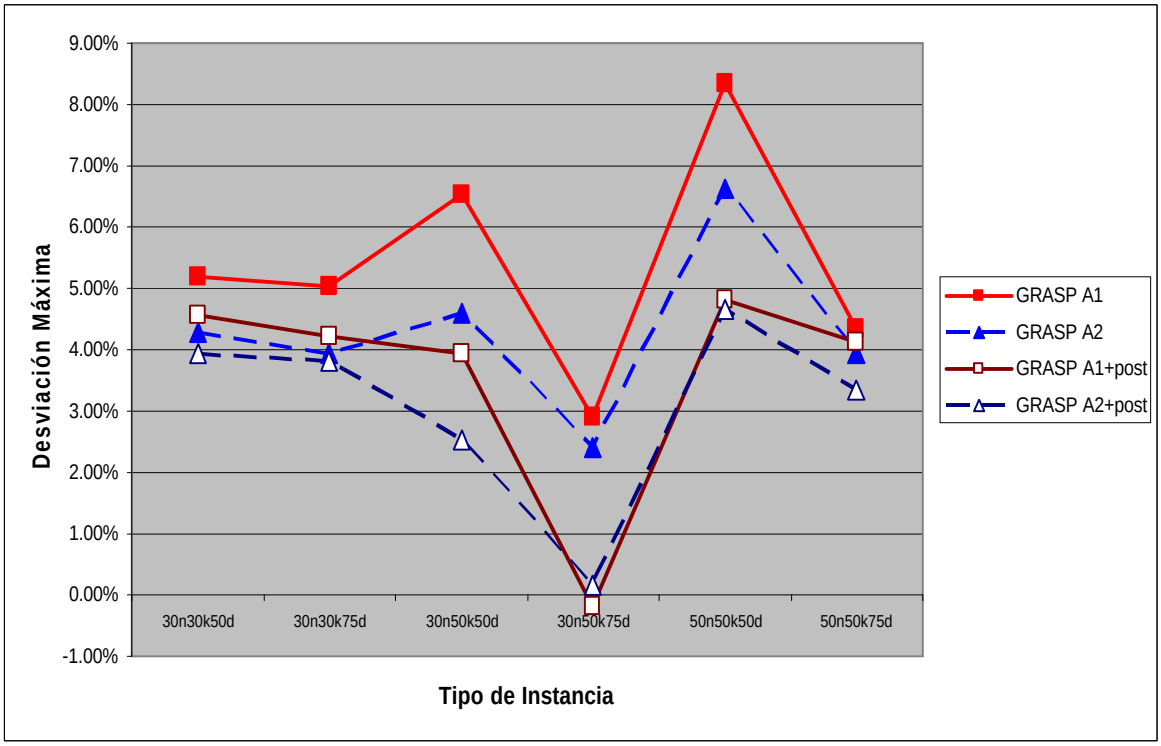
Gráfica 22. Desviación promedio de las instancias VT. Versiones GRASP antes de post-procesar.



Gráfica 23. Desviación promedio VT GRASP^{A} y post-procesamientos.



Gráfica 24. Desviación máxima VT para cada versión de GRASP. Versiones GRASP antes de post-procesar.



Gráfica 25. Desviación máxima VT GRASP^{A} y post-procesamientos.

CAPÍTULO 7. CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones.

Después de evaluado y analizado el rendimiento de las diversas versiones de GRASP que fueron diseñadas e implementadas para resolver el problema robusto de diseño de red capacitada multiproducto:

- $GRASP^{M}$
- $GRASP^{R}$
- $GRASP^{A}$

podemos llegar a varias conclusiones.

Entre lo más relevante debemos destacar que las versiones de $GRASP^{M}$ y Robusto tuvieron por regla general un rendimiento mediocre, sin mayores logros que acercarse medianamente a la otra versión de GRASP con buenos resultados: el $GRASP^{A}$.

Para las primeras versiones de GRASP, esto es, $GRASP^{M}$ y $GRASP^{R}$, no vale la pena dedicar mayor mención, a diferencia del $GRASP^{A}$ para quien se desarrolló un post-procesamiento que mejoraba las ya de por sí buenas soluciones. En las versiones de $GRASP^{M}$ y $GRASP^{R}$ solo se consideró la fase constructiva, dejando de lado la fase de post-procesamiento, la cual fue enfocada únicamente al $GRASP^{A}$, por mostrar mejores resultados.

El $GRASP^{A}$ entregó muy buenos resultados, mejorando aún más al incorporar el post-procesamiento. Una manera de encontrar soluciones de mayor calidad fue incrementar el número de iteraciones para dejar correr a $GRASP^{A}$ por más tiempo, logrando mejorar las soluciones entre un 2 y un 5% en promedio, pero con la inversión de 10 veces más en tiempo de cómputo.

Los tiempos de cálculo son bastante bajos, por lo general en el orden de los 5 minutos para 1,000 iteraciones y mostrando un comportamiento lineal en relación de aumento de iteraciones contra tiempo.

Comparando nuestro algoritmo con trabajos previos, siempre fue mejor al tratarse del GRASP^{A} con 10,000 iteraciones y post-procesamiento, en cuanto a tiempos de corrida (de magnitudes similares), así como en desviaciones promedio y máximas. La versión de 1,000 iteraciones entregó resultados comparables en cuanto a calidad de soluciones, pero en la décima parte del tiempo.

Adicionalmente nuestro algoritmo, en comparación al trabajo desarrollado anteriormente presenta mejoras en cuanto a que no tiene limitaciones para trabajar solo con instancias de ciertas características, como ocurre con el trabajo que nos sirve de comparación.

Un detalle interesante fue observado en el desempeño del post-procesamiento cuando era aplicado para problemas de costos variables relevantes, para los cuales por lo general no reportaba mejoras sustanciales con respecto a la solución antes de pre-procesar, cosa contraria a lo ocurrido con los problemas de costos fijos relevantes, para quienes se obtenían mejoras considerables. Esto puede deberse a que el post-procesamiento busca eliminar aristas, objetivo benéfico sobremanera para una instancia con costos fijos elevados, pero no muy importante cuando la mayor proporción de costo de la solución recae en los costos de transportación o variables. Sin embargo el desempeño total del pre-procesamiento mostró ser muy bueno.

Dejando al margen tecnicismos, el GRASP^{A} fue mejor porque busca enviar las demandas respectivas de cada producto para un solo escenario a la vez, con lo que se logran encontrar soluciones parcialmente construidas que al satisfacer las demandas de un escenario muy "saturado" de productos, satisfacen al mismo tiempo varios escenarios menos "saturados". ¿Qué conseguimos con esto? Encontrar diseños más ajustados a los diseños óptimos del problema determinístico que se desprende de cada escenario de nuestro problema, más económicos en términos de función objetivo, por tanto, mejores. En conclusión nuestro algoritmo es mucho mejor que el trabajo previo de Medina (2005 [61]), bajo igualdad de condiciones.

7.2 Recomendaciones.

Si bien nuestro algoritmo de GRASP^{A} entrega muy buenas soluciones, existen detalles importantes que vale la pena mencionar como recomendaciones de trabajos futuros y se enlistan a continuación:

1.- Clasificación de Instancias. Si bien clasificamos nuestras instancias con respecto a ciertas características descritas aquí, no existe una forma estándar o generalizada para clasificar las instancias de problemas de red. Cada autor maneja ciertas formas de decidir si una red es o no de cierta forma. Sería interesante proponer una forma estándar de identificar los problemas de red.

2.- Creación de una base de datos para problemas de diseño de red. No existen instancias de prueba para problemas de diseño de red, esta área sería muy útil para trabajar algoritmos que solucionen problemas de diseño y flujo en red, ya sea de manera determinística o bajo incertidumbre.

3.- Diferentes análisis de longitudes de rutas cortas. Dado que para construir nuestras soluciones debemos seleccionar rutas de una lista inicial, sería interesante determinar las rutas cortas iniciales de maneras distintas, incluso las rutas cortas encontradas en los puntos intermedios de nuestro algoritmo GRASP^{A}.

4.- Desarrollar cotas inferiores. Si bien las cotas superiores de buena calidad que empleamos fueron de gran ayuda, es todavía necesario disponer a cotas inferiores de nuestros problemas de diseño de red, ya que son las cotas inferiores y no las superiores quienes realmente pueden darnos índices más exactos de la calidad de una solución heurística.

5.- Otras consideraciones. El análisis de diferentes metodologías para propósito de comparación deberá ser un aspecto a considerar para diferentes trabajos futuros. Los métodos heurísticos se encuentran en un punto de desarrollo tal que es necesario diferentes estudios para poder decidir si una determinada metodología es preferible a otra para algún tipo de problema particular.

REFERENCIAS BIBLIOGRÁFICAS

[1] Arráiz, E.; Martínez, A.; Meza, O.; Ortega, M. *GRASP and Tabu Search algorithms for computing the forwarding index in a graph.* En Proceedings of MIC 2001, pp. 367-370. Porto, Portugal, Julio, 2001.

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

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

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

[5] Binato, S.; Hery, W.J.; Loewenstern, D.; Resende, M.G.C. *A greedy randomized adaptive search procedure for job shop scheduling.* En Ribeiro, C. y Hansen, P. (editores), Essays and Surveys on Metaheuristics, pp. 58-79. Kluwer, Boston, EUA, 2002.

[6] Chardaire, P.; McKeown, G.P.; Maki, J.A. *Application of GRASP to the multiconstraint knapsack problem.* En Boers, E.J.W.; Gottlieb, J.; Lanzi, P.L.; Smith, R.E.; Cagnoni, S.; Hart, E. ; Raidl, G.R. y Tijink, H. (editores), Applications of Evolutionary Computing, Lecture Notes in Computer Science, Vol. 2037, pp. 30-39. Springer, Heidelberg, Alemania, 2001.

[7] Cobos-Zaleta, N. *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, NL, México, Julio 2004.

[8] Cobos-Zaleta, N.; Álvarez, A.M. *Tabu Search-based algorithm for Capacitated Multicommodity Network Design Problem*. Proceedings of the XIV International

Conference on Electronics, Communications and Computers. Veracruz, México, pp. 144-148, 2004.

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

[10] Crainic, T.G. *Service network design in freight transportation.* European Journal of Operational Research, Vol. 122, No. 2, pp. 272-288, 2000.

[11] Crainic, T.G.; Frangioni, A.; Gendron, B. *Bundle-based relaxation methods for multicommodity capacitated fixed charge network design.* Discrete Applied Mathematics, Vol.112, No.1-3, pp. 73-79, 2001.

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

[13] Cruz, F.R.B.; MacGregor, J.; Mateus, G.R. *Solving to optimality the uncapacitated fixed-charge network flow problem.* Computers and Operations Research, Vol. 25, No. 1, pp. 67-81, 1998.

[14] Darlington, J.; Pantelides, C.C.; Rustem, B.; Tanyi, B.A. *Decreasing the sensitivity of open-loop optimal solutions in decision making under uncertainty.* European Journal of Operational Research, Vol. 121, pp. 343–362. 2000.

[15] Daskin, M.S. *Network and discrete location. Models, Algorithms and Applications.* Wiley. New York, EUA, 1995.

[16] De Alba, K. *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, N.L. México, Febrero 2004.

[17] De Alba, K.; Álvarez, A.M.; González-Velarde, J.L. *A memory-based GRASP for finding good starting solutions to the multicommodity capacitated network design problem*. Proceedings, INFORMS Computing Society's Conference. Chandler, Arizona, EUA. Enero 2003.

[18] De Alba, K.; Álvarez, A.M.; González-Velarde, J.L. *Un algoritmo de búsqueda para un problema de red capacitada multiproducto*. Memorias del 3er. Encuentro Internacional de Computación. Aguascalientes, Ags. Septiembre 2001.

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

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

[21] Eppen, G.D.; Kipp, M.; Schrage, L. *A scenario approach to capacity planning*. Operations Research, Vol. 37, No. 4, pp. 517-528, July-August 1989.

[22] Erickson, R.E.; Monma, C.L. y Veinott, A.F. *Send-and-split methods for minimum concave cost network flows.* Mathematics of Operations Research, Vol. 12, No. 4, pp. 634-664, 1987.

[23] Feo, T.A.; Resende, M.G.C. *A probabilistic heuristic for a computationally difficult set covering problem.* Operations Research Letters, Vol. 8, No. 2, pp. 67-71, 1989.

[24] Fisher, M.L.; Jaikumer, R. *A decomposition algorithm for large scale vehicle routing*. Technical Report 78-11-05, Department of Decision Sciences, University of Pennsylvania, 1978.

[25] Fleurent, C.; Glover, F. *Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory.* INFORMS Journal on Computing. Vol. 11, No. 2, pp. 198-204, 1999.

[26] Florian, M.G.; Guerin, G.; Bushel, G. *The engine scheduling problem in a railway network.* INFOR Journal, Vol. 14, pp. 121–138, 1976.

[27] Gabow, H.N.; Goemans, M.X.; Williamson, D.P. *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.

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

[29] Gendron, B. A note on *A dual-ascent approach to the fixed-charge capacitated network design problems.* European Journal of Operational Research, Vol.138, No.3, pp. 671-675, 2002.

[30] Gendron, B.; Crainic, T.G. *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.

[31] Geoffrion, A.M.; Graves, G.W. *Multicommodity distribution system design by Benders decomposition*. Management Science, Vol. 20, pp. 822–844, 1974.

[32] Glover, F.; Laguna, M. *Tabu Search.* Kluwer Academic Publisher, Norwell, MA, 1997.

[33] Glover, F.; Laguna, M.; Martí, R. *Scatter search and path relinking: Advances and applications.* En Glover, F. y Kochenberger, G. (editores), Handbook of Metaheuristics. pp. 1-35. Kluwer, Boston, EUA, 2003.

[34] Gutiérrez, G.J. ; Kouvelis, P. *A robustness approach to international sourcing*. Annals of Operations Research, Vol. 59, pp. 165–193. 1995.

[35] Gutiérrez, G.J.; Kouvelis, P.; Kurawarwala, A.A. *A Robustness Approach to Uncapacitated Network Design Problems.* European Journal of Operational Research, Vol. 94, No. 2, pp.362-376, 1996.

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

[37] Herrmann, J.W. *A Genetic Algorithm for a Minimax Network Design Problem.* Proceedings of the 1999 Congress on Evolutionary Computation. Washington, D.C., pp. 1099-1103, July 6-9, 1999.

[38] Herrmann, J.W.; Ioannou, G.; Minis, I.; Proth, J.M. *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.

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

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

[41] Holmberg, K.; Hellstrand, J. *Solving the uncapacitated network design problem by a Lagrangean heuristic and branch-and-bound*. Operations Research, Vol. 46, No. 2, pp. 247-258, 1998.

[42] Holmberg, K.; Migdalas, A. *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.

[43] Holmberg, K.; Yuan, D. *A Lagrangean approach to network design problems.* Int. Trans. Operational Research. Vol. 5, No.6, pp. 529-539, 1998.

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

[45] Johnson, D.S.; Lenstra, J.K.; Rinnooy, H.G. *The complexity of the network design problem.* Networks, Vol. 8, No. 4, pp. 279-285, 1978.

[46] Kennington, J.; Lewis, K.; Olinick, E.; Ortynsky, A.; Spiride, G. *Robust Solutions for the WDM Routing and Provisioning Problem: Models and Algorithms.* Optical Network Magazine, Vol. 4, pp. 74–84, 2003.

[47] Khang, D.B.; Fujiwara, O. *Approximate solutions of capacitated fixed charge minimum cost network flow problems.* Networks, Vol. 21, pp. 689-704, 1991.

[48] Killmer, K.A.; Anandalingam, G.; Malcolm, S.A. *Siting noxious facilities under uncertainty.* European Journal of Operational Research, Vol. 133, pp. 596–607. 2001.

[49] Kouvelis, P. ; Kurawarwala, A.A.; Gutiérrez, G.J. *Algorithms for robust single and multiple period layout planning for manufacturing systems.* European Journal of Operational Research, Vol. 63, pp. 287–303, 1992.

[50] Kouvelis, P. ; Yu, G. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, 1997.

[51] Laguna, M. Applying Robust *Optimization to Capacity Expansion of One Location in Telecommunications with Demand Uncertainty*. Management Science, Vol. 44, no. 11, pp. S101-S110, 1997.

[52] Laguna, M.; González-Velarde, J.L. *A Benders-based Heuristic for the Robust Capacitated Internacional Sourcing Problem.* IIE Transactions, Vol. 36, pp. 1125-1133, 2004

[53] Laguna, M.; Lino, P.; Pérez, A.; Quintanilla, S.; Valls, V. *Minimizing weighted tardiness of jobs with stochastic interruptions in parallel machines.* European Journal of Operational Research 127, pp. 444–457, 2000.

[54] Lawler, E.L. *A procedure for computing the k-th best solutions to discrete optimization problems and its applications to the shortest path problem.* Managment Science, Vol. 18, pp. 401-405, 1972.

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

[56] Magnanti, T.L.; Mirchandani, P.; Wong, R.T. *Tailoring Benders decomposition for uncapacitated network design.* Mathematical Programming Study, Vol. 26, pp. 112–154, 1986.

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

[58] Mausser, H.E.; Laguna, M. *A Heuristic to minimax absolute regret for linear programs with interval objective function coefficients.* European Journal of Operational Research, Vol. 117, pp. 157-174, 1999a.

[59] Mausser, H.E.; Laguna, M. *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. 1999b.

[60] Medina, J. F.A.; Álvarez, A.M.; De Alba K. *A Heuristic to Find a P% Robust Network Design.* WSEAS Transacions on Circuits and Systems, Vol. 3, pp. 2247-2253, 2004.

[61] Medina, J.F.A. *Un enfoque robusto a un problema de Diseño de Red multiproducto con incertidumbre en parámetros de entrada.* Tesis de Maestría. FIME, UANL. San Nicolás de los Garza, NL, México, Junio 2005.

[62] Mulvey, J.M.; Vanderbei, R.J.; Zenios, S.A. *Robust Optimization of Large-Scale Systems.* Operations Research, Vol. 43, pp. 264-281, 1995.

[63] Osman, I.H.; Kelly, J.P. *Metaheuristics: Theory and applications*. Kluwer Academic Publishers, Norwell, MA. 1996.

[64] Raja, V.T.; Han, B.T. *A Scenario-based Robust Optimization Model for Optimal Design of Local Access Network*, under review for possible publication in Telecommunication Systems Journal.

[65] Richardson, R. *An optimization approach to routing aircraft.* Transportation Science. Vol. 10, pp. 52–71, 1976.

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

[67] Snyder, L.V.; Daskin, M.S. *Stochastic p-robust location problems.* Trabajo de investigación en desarrollo, Northwestern University, 2003.

[68] Taillard, E.; Gambardella, L.; Gendreau, M.; Potvin, J. *Adaptive memory programming: "A unified view of metaheuristics"*. European Journal of Operational Research, Vol. 135, No. 1, pp. 1-16, 2001.

[69] Trafalis, T.B.; Mishina, T.; Foote, B.L. *An interior point multiobjective programming approach for production planning with uncertain information.* Computers & Industrial Engineering, Vol. 37, pp. 631–648, 1999.

[70] Yu, C.S.; Li, L.H. *A robust optimization model for stochastic logistic problems.* International Journal of Production Economics, Vol. 64, pp. 385–397, 2000.

[71] Zanakis, S.H.; Evans, J.R. *Heuristic optimization: Why, when, and how to use it.* Interfaces, Vol. 11, No. 5, pp. 84-91, 1981.

ÍNDICE DE FIGURAS, TABLAS Y GRÁFICAS

FIGURAS:

TABLAS:

GRÁFICAS:

ANEXOS

Anexo A1: Ajustando Lambda, su Incremento y Decremento

Parámetros	Desviación Promedio (Lambdas, L+, L-)	Máxima Desviación (Lambdas, L+, L-)
01 02 02	1.91%	5.44%
01 02 05	2.32%	5.88%
01 02 25	3.54%	7.13%
01 05 02	**1.84%**	**5.38%**
01 05 05	1.97%	5.46%
01 05 25	2.78%	6.38%
01 25 02	1.85%	5.38%
01 25 05	1.92%	5.43%
01 25 25	2.11%	5.67%
10 02 02	2.14%	5.79%
10 02 05	2.40%	6.00%
10 02 25	2.85%	6.48%
10 05 02	1.90%	5.48%
10 05 05	1.93%	5.51%
10 05 25	2.58%	6.20%
10 25 02	1.90%	5.47%
10 25 05	2.13%	5.63%
10 25 25	2.20%	5.82%

Anexo A2: Ajustando Parámetro de Sustitución en Conjunto Élite y Diversidad de Población

Parámetros	Desviación Promedio (us, um)	Máxima Desviación (su, sm)
01 05 02 10 10 10	1.49%	5.00%
01 05 02 10 30 10	1.92%	5.39%
01 05 02 10 35 10	2.10%	5.66%
01 05 02 10 60 10	2.86%	6.49%
01 05 02 30 10 10	1.73%	5.25%
01 05 02 30 30 10	1.84%	5.38%
01 05 02 30 35 10	2.24%	5.82%
01 05 02 30 60 10	3.00%	6.61%
01 05 02 60 10 10	3.29%	6.90%
01 05 02 60 30 10	3.50%	7.09%
01 05 02 60 35 10	3.74%	7.39%
01 05 02 60 60 10	4.79%	8.48%
10 05 02 10 10 10	**1.31%**	**4.92%**
10 05 02 10 30 10	1.75%	5.36%
10 05 02 10 35 10	1.84%	5.41%

10 05 02 10 60 10	2.35%	5.89%
10 05 02 30 10 10	1.71%	5.23%
10 05 02 30 30 10	1.90%	5.48%
10 05 02 30 35 10	1.86%	5.43%
10 05 02 30 60 10	2.55%	6.14%
10 05 02 60 10 10	4.21%	7.95%
10 05 02 60 30 10	4.46%	8.18%
10 05 02 60 35 10	4.59%	8.34%
10 05 02 60 60 10	5.09%	8.93%

Anexo A3: Ajustando Longitud de LRC

Parámetros	Desviación Promedio (Long. LRC)	Máxima Desviación (Long. LRC)
01 05 02 30 30 10	**1.84%**	**5.38%**
01 05 02 30 35 10	2.24%	5.82%
01 05 02 30 30 20	4.10%	7.74%
01 05 02 30 35 20	4.76%	8.38%
01 05 02 30 30 30	7.55%	11.31%
01 05 02 30 35 30	8.01%	11.87%
01 05 02 30 30 60	18.39%	22.79%
01 05 02 30 35 60	18.73%	23.11%
10 05 02 30 30 10	1.90%	5.48%
10 05 02 30 35 10	1.86%	5.43%
10 05 02 30 30 20	4.40%	8.05%
10 05 02 30 35 20	4.70%	8.33%
10 05 02 30 30 30	7.74%	11.56%
10 05 02 30 35 30	8.35%	12.16%
10 05 02 30 30 60	18.81%	23.12%
10 05 02 30 35 60	18.95%	23.32%

Anexo A4: Tiempo (segs.)

1,000 iteraciones

	fl	ft	vl	vt	Prom
30n30k50d	69.60	69.20	66.20	67.40	68.10
30n30k75d	66.00	82.40	61.60	77.00	71.75
30n50k50d	76.00	94.80	71.20	84.00	81.50
30n50k75d	75.60	136.00	66.80	115.60	98.50
50n50k50d	79.20	231.00	71.20	158.60	135.00
50n50k75d	94.20	255.60	77.60	138.00	141.35

Anexo A5: Tiempo (segs.)

10,000 iteraciones

	fl	ft	vl	vt	Prom
30n30k50d	516.40	573.80	519.20	559.80	542.30
30n30k75d	549.00	699.00	516.40	672.40	609.20
30n50k50d	715.00	983.20	721.00	901.80	830.25
30n50k75d	627.00	1282.20	552.00	1212.60	918.45

50n50k50d	702.60	2252.00	621.40	1725.20	1325.30
50n50k75d	788.00	2313.60	659.20	1452.20	1303.25

Anexo A6: Desviación Promedio

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1+post**	**grasp A2**	**grasp A2+post**
30n30k50d	5.77%	7.91%	8.50%	2.40%	0.15%	0.47%	-1.83%
30n30k75d	5.77%	5.99%	4.59%	-0.76%	-3.52%	-2.18%	-3.95%
30n50k50d	0.00%	14.70%	13.00%	3.54%	-3.71%	0.43%	-5.57%
30n50k75d	5.19%	14.85%	14.70%	-0.14%	-7.09%	-2.56%	-8.12%
50n50k50d	0.00%	11.48%	10.50%	4.51%	-0.61%	1.70%	-2.10%
50n50k75d	-2.31%	4.72%	3.46%	1.59%	-1.90%	0.39%	-2.51%

Anexo A7: Desviación Máxima

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	7.83%	16.64%	15.92%	5.65%	3.54%	3.67%	1.49%
30n30k75d	1.57%	14.08%	12.15%	3.44%	0.67%	1.76%	-0.02%
30n50k50d	-5.76%	23.69%	21.20%	6.80%	-0.80%	3.55%	-2.59%
30n50k75d	-7.14%	23.64%	23.25%	3.56%	-3.62%	0.96%	-4.74%
50n50k50d	6.02%	19.77%	18.70%	8.19%	2.87%	5.13%	1.15%
50n50k75d	2.77%	11.38%	10.05%	4.91%	1.58%	3.80%	0.90%

Anexo A8: Problemas Tipo FL. Desviación Promedio (Costos Fijos Relevantes, Capacidad Holgada)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	2.35%	6.56%	6.40%	2.39%	-2.79%	-0.67%	-5.39%
30n30k75d	-2.31%	7.77%	7.33%	-1.89%	-5.93%	-4.42%	-7.45%
30n50k50d	-11.41%	16.11%	12.22%	4.55%	-9.93%	0.33%	-11.56%
30n50k75d	-15.20%	6.84%	6.75%	-2.64%	-14.61%	-6.75%	-17.15%
50n50k50d	0.05%	9.26%	6.93%	3.15%	-6.30%	-1.15%	-7.85%
50n50k75d	-0.54%	7.31%	7.51%	2.30%	-4.33%	-0.48%	-4.82%

Anexo A9: Problemas Tipo FL Desviación Máxima (Costos Fijos Relevantes, Capacidad Holgada)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	5.19%	16.30%	13.44%	4.95%	-0.17%	1.77%	-2.96%
30n30k75d	2.21%	16.52%	17.09%	2.62%	-1.60%	-0.20%	-3.31%
30n50k50d	-8.37%	24.32%	19.58%	8.18%	-6.87%	3.73%	-8.50%
30n50k75d	-11.20%	16.16%	18.05%	2.08%	-10.47%	-2.21%	-13.13%
50n50k50d	4.09%	17.54%	13.48%	7.24%	-2.52%	2.75%	-4.32%
50n50k75d	2.19%	13.55%	11.98%	5.11%	-1.68%	2.25%	-2.12%

Anexo A10: Problemas Tipo FT Desviación Promedio (Costos Fijos Relevantes, Capacidad Justa)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	5.77%	15.09%	16.63%	5.16%	2.72%	2.64%	-1.06%
30n30k75d	-4.55%	3.02%	2.91%	-1.51%	-6.73%	-3.48%	-6.83%
30n50k50d	-6.51%	20.99%	17.26%	4.82%	-4.37%	1.07%	-7.75%
30n50k75d	-7.45%	23.08%	24.94%	3.34%	-6.89%	-0.70%	-8.43%
50n50k50d	3.27%	19.22%	18.32%	7.16%	0.78%	4.63%	-0.91%
50n50k75d	-2.90%	3.07%	-0.97%	1.76%	-4.34%	0.71%	-4.96%

Anexo A11: Problemas Tipo FT Desviación Máxima (Costos Fijos Relevantes, Capacidad Justa)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	10.46%	25.12%	25.59%	9.70%	7.16%	7.11%	3.23%
30n30k75d	0.93%	12.77%	11.13%	4.35%	-1.19%	2.25%	-1.33%
30n50k50d	-3.16%	30.82%	28.02%	8.55%	-0.97%	4.61%	-4.53%
30n50k75d	-3.08%	34.19%	35.99%	8.44%	-2.50%	4.09%	-3.99%
50n50k50d	7.94%	33.34%	34.08%	12.26%	5.47%	9.53%	3.68%
50n50k75d	3.36%	12.94%	10.96%	8.28%	1.99%	7.25%	1.29%

Anexo A12: Problemas Tipo VL Desviación Promedio (Costos Variables Relevantes, Capacidad Holgada)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	-	2.89%	3.87%	-0.15%	-0.71%	-1.04%	-1.36%
30n30k75d	-	7.77%	3.45%	-0.48%	-1.37%	-1.19%	-1.57%
30n50k50d	-	8.53%	8.52%	1.45%	-1.34%	-0.90%	-2.20%
30n50k75d	-	6.82%	7.35%	-1.24%	-3.98%	-2.41%	-4.18%
50n50k50d	-	8.02%	6.30%	1.83%	0.79%	-0.63%	-1.52%
50n50k75d	-	6.20%	4.69%	0.08%	-0.41%	-0.41%	-0.99%

Anexo A13: Problemas Tipo VL Desviación Máxima (Costos Variables Relevantes, Capacidad Holgada)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	-	10.76%	8.58%	2.75%	2.61%	1.52%	1.78%
30n30k75d	-	14.22%	9.40%	1.76%	1.25%	1.04%	0.76%
30n50k50d	-	18.36%	16.52%	3.95%	0.69%	1.29%	0.14%
30n50k75d	-	13.21%	13.02%	0.80%	-1.31%	-0.45%	-2.01%
50n50k50d	-	14.35%	10.81%	4.92%	3.73%	1.63%	0.57%
50n50k75d	-	13.12%	10.85%	1.89%	1.89%	1.74%	1.09%

Anexo A14: Problemas Tipo VT Desviación Promedio (Costos Variables Relevantes, Capacidad Justa)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	-	7.11%	7.10%	2.21%	1.38%	0.95%	0.51%
30n30k75d	-	5.42%	4.66%	0.84%	-0.04%	0.35%	0.04%
30n50k50d	-	13.19%	14.01%	3.32%	0.82%	1.20%	-0.77%
30n50k75d	-	22.67%	19.75%	-0.01%	-2.87%	-0.40%	-2.71%
50n50k50d	-	9.41%	10.45%	5.92%	2.30%	3.97%	1.88%
50n50k75d	-	2.31%	2.60%	2.21%	1.48%	1.74%	0.73%

Anexo A15: Problemas Tipo VT Desviación Máxima (Costos Variables Relevantes, Capacidad Justa)

		1,000 iteraciones				*10,000 iteraciones*	
Problema	**Trabajo Previo**	**grasp M**	**grasp R**	**grasp A1**	**grasp A1 + post**	**grasp A2**	**grasp A2 + post**
30n30k50d	-	14.38%	16.07%	5.20%	4.55%	4.29%	3.93%
30n30k75d	-	12.79%	10.97%	5.05%	4.22%	3.94%	3.80%
30n50k50d	-	21.24%	20.66%	6.52%	3.93%	4.60%	2.52%
30n50k75d	-	30.99%	25.94%	2.91%	-0.18%	2.41%	0.15%
50n50k50d	-	13.86%	16.44%	8.34%	4.81%	6.62%	4.67%
50n50k75d	-	5.92%	6.40%	4.35%	4.13%	3.95%	3.35%

RESUMEN AUTOBIOGRÁFICO

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

Tesis:

HEURÍSTICA PARA RESOLVER EL PROBLEMA ROBUSTO DE DISEÑO DE RED CAPACITADA MULTIPRODUCTO

Nacido en Monterrey, Nuevo León, México. Es hijo del Ing. Fernando Pérez Chávez y Julia A. González de Pérez. Recibió el título de Ingeniero Mecánico Administrador en la Universidad Autónoma de Nuevo Léon en el año 2000. Actualmente labora en PROLEC GE en la ciudad de Apodaca Nuevo León.

Los resultados del presente trabajo han sido publicados en:

"Pre-processing a Network Design Problem using GRASP".
Fernando Pérez González, Ada M. Álvarez S., Karim De Alba R.
WSEAS Transactions on Mathematics. Issue 2, Volume 4,
April 2005, pp 95 – 101. ISSN 1109-2769

"A Constructive Procedure for Finding Good Starting Solutions to the Network Design Problem with Uncertain Parameters".
Fernando Pérez González, Ada M. Álvarez S., Karim De Alba R.
Proceedings of the WSEAS International Conferences ESPOCO '05, AMCOS '05, ICOSSE '05, ICOPES '05. Río de Janeiro, Brazil. April 25-27, 2005

y presentados en:

“Problema Robusto de Diseño de Red Capacitada Multiproducto”.
Ciclo de Seminarios de Investigación del Programa de Posgrado en Ingeniería de Sistemas. San Nicolás de los Garza, Nuevo León, México.
24 de mayo del 2004.

“Generando Soluciones Iniciales para el Problema de Diseño Robusto de Red Capacitada Multiproducto”.
Ciclo de Seminarios de Investigación del Programa de Posgrado en Ingeniería de Sistemas. San Nicolás de los Garza, Nuevo León, México.
15 de noviembre del 2004.

“GRASP para Solucionar un Problema de Diseño Robusto de Red Capacitada Multiproducto”.
XV Escuela Nacional de Optimización y Análisis Numérico (ENOAN). Morelia, Michoacán, México.
20 de abril del 2005.